

CPU vs GPU

How it matters?

by Laurent Magnin, PhD

MGL 7320: Ingénierie logicielle des systèmes d'intelligence artificielle

EP-MEANS: An Efficient Nonparametric Clustering of Empirical Probability Distributions

2.2 Preliminaries

Earth Mover's Distance (EMD) is a dissimilarity metric that meets all the requirements in Section 1. Given two probability distributions P and Q , the EMD [10] is most easily understood as the total area between their *cumulative distribution functions* (CDFs). Recall that the CDF of a distribution is a nondecreasing function $CDF(x)$ whose value at any real number x is the probability that a draw from the distribution will be less than or equal to x .

$$EMD(P, Q) = \int_{x=0}^1 |CDF_P^{-1}(x) - CDF_Q^{-1}(x)| \quad (1)$$

We use Equation 1 to compute the EMD between two distributions. If we consider the two distributions as piles of earth, then EMD computes the minimum total distance that earth must be moved to transform one into the other.

We use ***k*-means clustering** with EMD as the distance metric to cluster distributions. We describe model selection and initialization in Section 3.

3.2 Centroid Computation

Given a collection of distributions $\{P_1 \dots P_m\}$, their centroid for the purposes of *k*-means clustering is another empirical distribution Q such that $\sum_{i=1}^m (EMD(P_i, Q))^2$ is minimized. The intuition behind computing Q is based on Equation 1. If we consider the CDFs of all P_i 's simultaneously, the value of CDF_Q at any given height y is just the mean value of all P_i 's at height y .

To illustrate this further, let $r_i = \min(\{v | P_i(v) > 0\})$ and $s = \min(P_i(r_i))$. So for $0 \leq x < s$, $CDF_{P_i}^{-1}(x) = r_i$. The inverse CDF of Q should also be constant on this interval. If $CDF_Q^{-1}(x) = y$ for $0 \leq x < s$, then the contribution of this interval to $EMD(P_i, Q)$ is just $s(r_i - y)$ by Equation 1. The total contribution of this interval to within-cluster squared error is $s^2 \sum_{i=1}^m (r_i - y)^2$, which is minimized by choosing $y = \frac{1}{m} \sum_{i=1}^m r_i$.

Applying this logic to the rest of the probability axis, we see that Q should be selected such that $CDF_Q^{-1}(x) =$

Code

766906d

Go to file

- whitepaperimages
 - DataCleanupAndExport.ipynb
 - EDA.ipynb
 - Flight Analysis with EP-Means.ipynb
 - Milestone1-Proposal-EP-Means.d...
 - Milestone2.md
 - Milestone2.pdf
 - Milestone3.md
 - Milestone3.pdf
 - Milestone4-10Questions.md
 - Milestone4-10Questions.pdf
 - Milestone4.md
 - Milestone4.pdf
 - Milestone4Presentation.pptx
 - README.md
 - henderson-sac15.pdf

Documentation • Share feedback

EPMeansFlights / Flight Analysis with EP-Means.ipynb

Preview Code Blame 936 lines (936 loc) · 89.9 KB

```
i = i + 1
Probability_Distribution_List.append(Probability_Distribution(df, route, "ACTUAL_ELAPSED_TIME"))
clear_output(wait=True)

2077 of 2077: HRL --> IAH

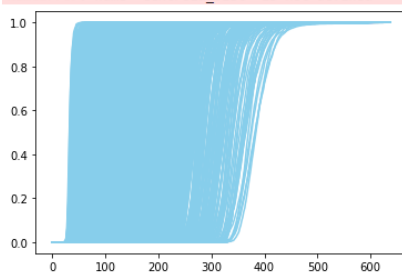
In [17]: # for i in range(5):
#         for j in range(5):
#             EMD = EarthMoversDistance(Probability_Distribution_List[i],Probability_Distribution_List[j])
#             print(f"EMD({i},{j}) = {EMD}")

In [18]: # Centroid_Dist = CentroidDistribution(Probability_Distribution_List)

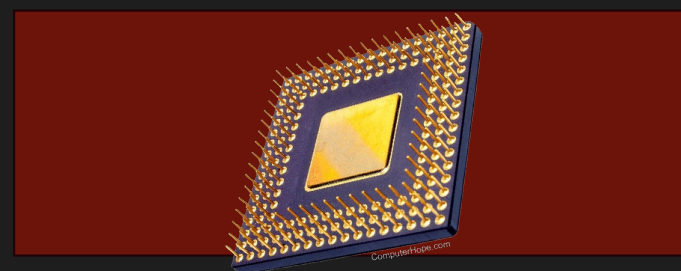
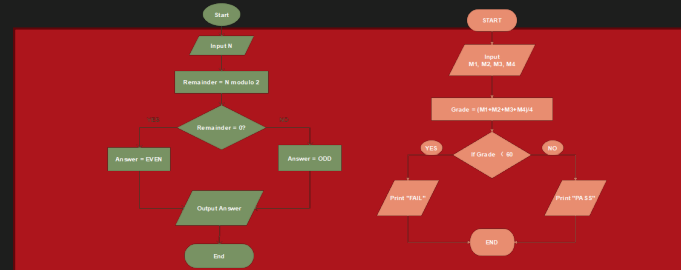
In [19]: # for P_Dist in Probability_Distribution_List:
#         plt.plot(range(P_Dist.max_field_value + 1),P_Dist.CDF,color='skyblue')
#         plt.plot(range(Centroid_Dist.max_field_value + 1),Centroid_Dist.CDF,color = 'black')
#         plt.show()

-----
NameError                                Traceback (most recent call last)
<ipython-input-19-ac7eb13cb762> in <module>
      1 for P_Dist in Probability_Distribution_List:
      2     plt.plot(range(P_Dist.max_field_value + 1),P_Dist.CDF,color='skyblue')
----> 3     plt.plot(range(Centroid_Dist.max_field_value + 1),Centroid_Dist.CDF,color = 'black')
      4     plt.show()

NameError: name 'Centroid_Dist' is not defined
```



```
In [20]: def GetMinEMDSquared(P_Dist,ClusterCenterList):
EMD_Squared_List = []
for ClusterCenter in ClusterCenterList:
    EMD_Squared_List.append(EarthMoversDistance(P_Dist, ClusterCenter) ** 2)
```



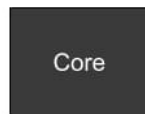
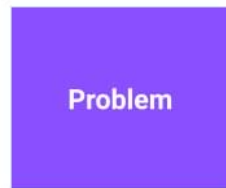
```
[72]: if profile:
      %lprun -f generate_probability_distribution_list -f Probability_Distribution.__init__ generate_probability_distribution_list(routes, "Name", "Distance")
```

```
Null Value
Null Value
Timer unit: 1e-06 s
```

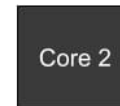
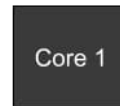
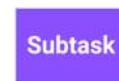
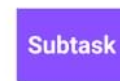
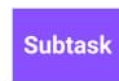
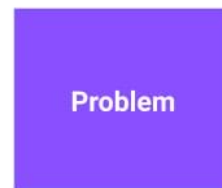
```
Total time: 3.70808 s
File: /tmp/ipykernel_8884/2400033246.py
Function: generate_probability_distribution_list at line 1
```

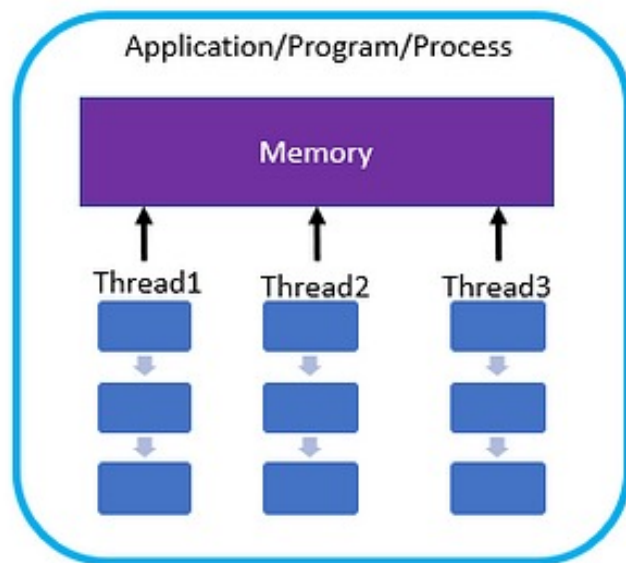
Line #	Hits	Time	Per Hit	% Time	Line Contents
1					def generate_probability_distribution_list(dataframe, individual_field, value_field):
2	1	3.0	3.0	0.0	Probability_Distribution_List = []
3	1	2.0	2.0	0.0	i = 1
4	1	11451.0	11451.0	0.3	sample = dataframe[individual_field].unique() #[:30]
5	1	4.0	4.0	0.0	last = len(sample)
6	693	545.0	0.8	0.0	for individual in sample:
7					# print(f"{i} of {last}: {individual}")
8	692	375.0	0.5	0.0	i = i + 1
9	692	283.0	0.4	0.0	try:
10	692	3695088.0	5339.7	99.6	Probability_Distribution_List.append(Probability_Distribution(dataframe, individual, individual_field, value_field))
11	2	1.0	0.5	0.0	except:
12	2	326.0	163.0	0.0	print("Null Value")
13					# clear_output(wait=True)
14					
15	1	0.0	0.0	0.0	return Probability_Distribution_List

Serial

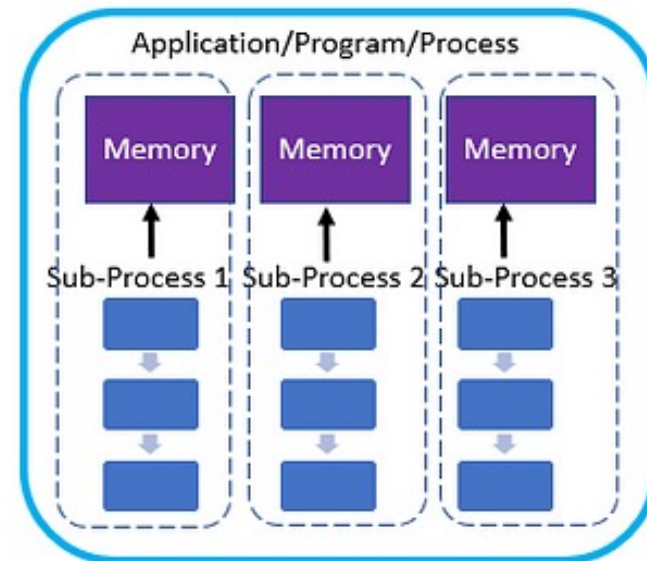


Parallel

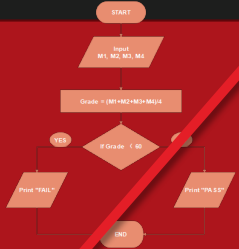
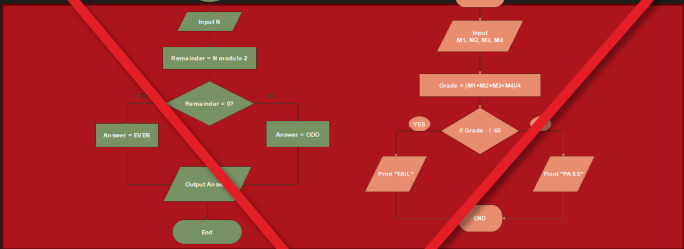




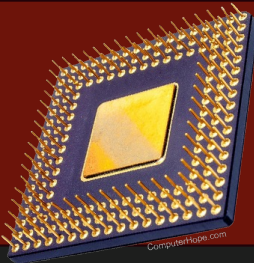
Multi-Threading

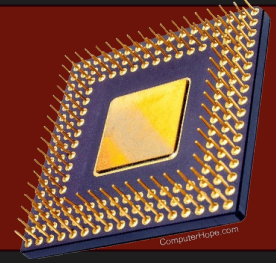
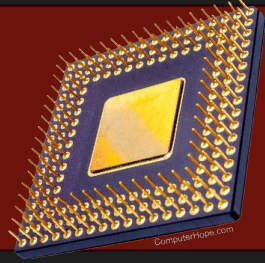
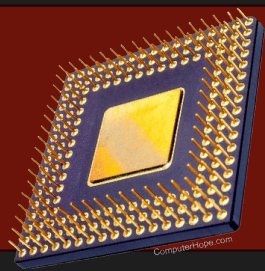
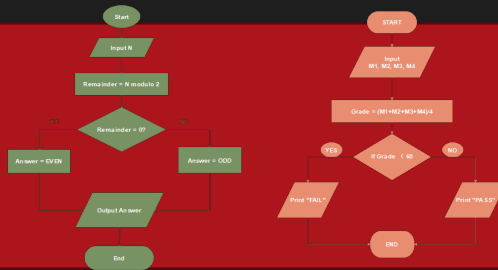


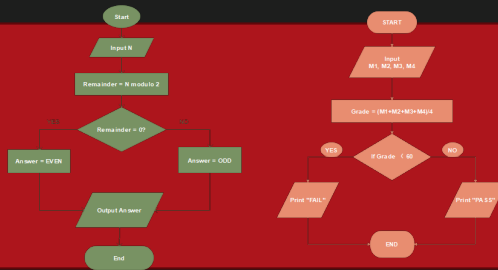
Multi-processing



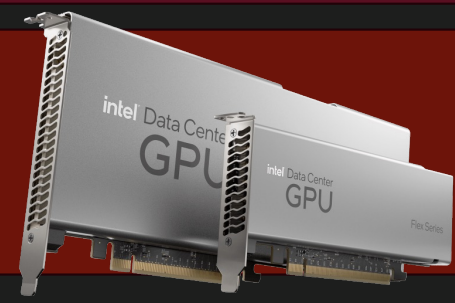
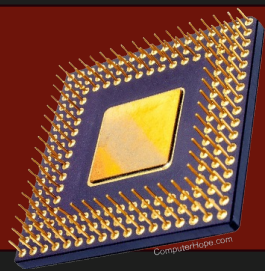
python



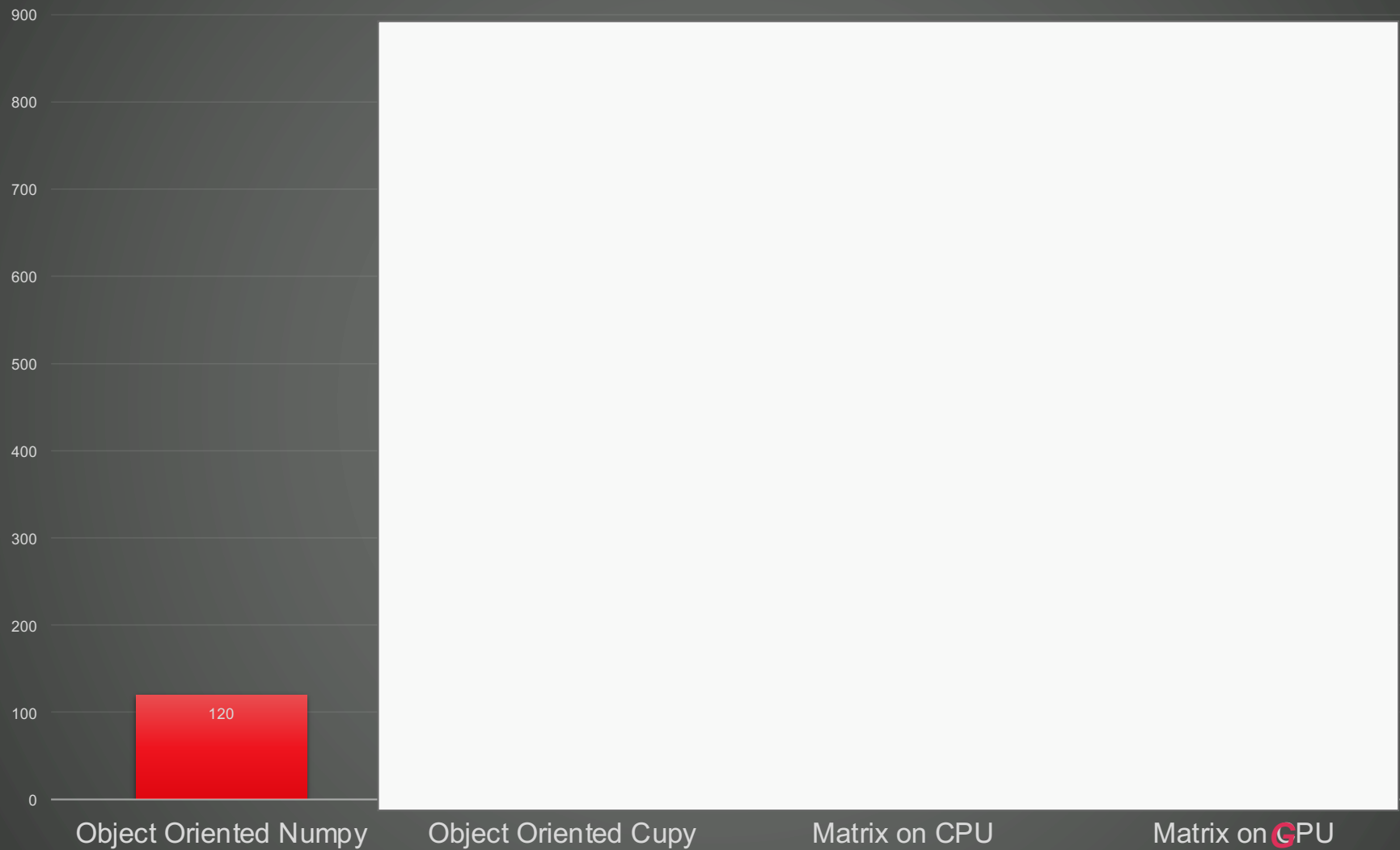




CuPy



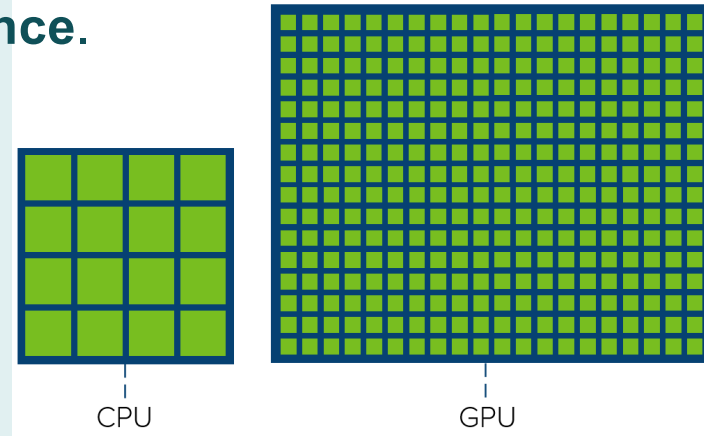
EPMeans, k=7. In seconds



CPU vs GPU – Generalities

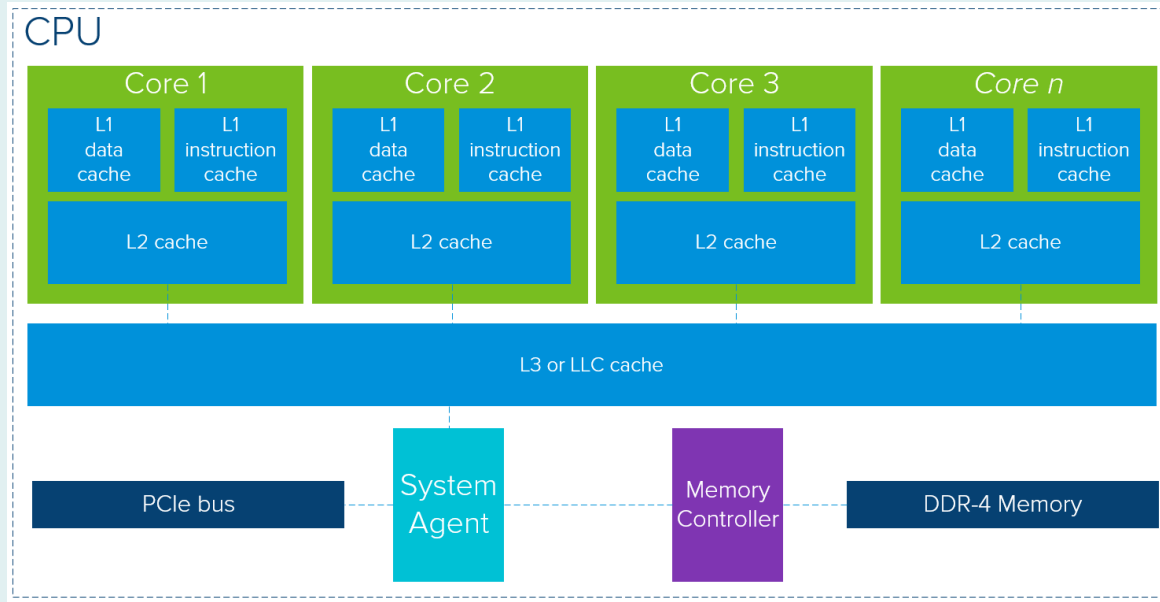
Different Processing Architectures – Latency vs Throughput

- A CPU is optimized to be as quick as possible to **finish a task at a as low as possible latency**, while keeping the ability to quickly switch between operations. Its nature is all about processing tasks in a **serialized way**.
- A GPU is all about **throughput optimization**, allowing to push as many as possible tasks through its internals **at once**.



[Exploring the GPU Architecture | VMware](#)

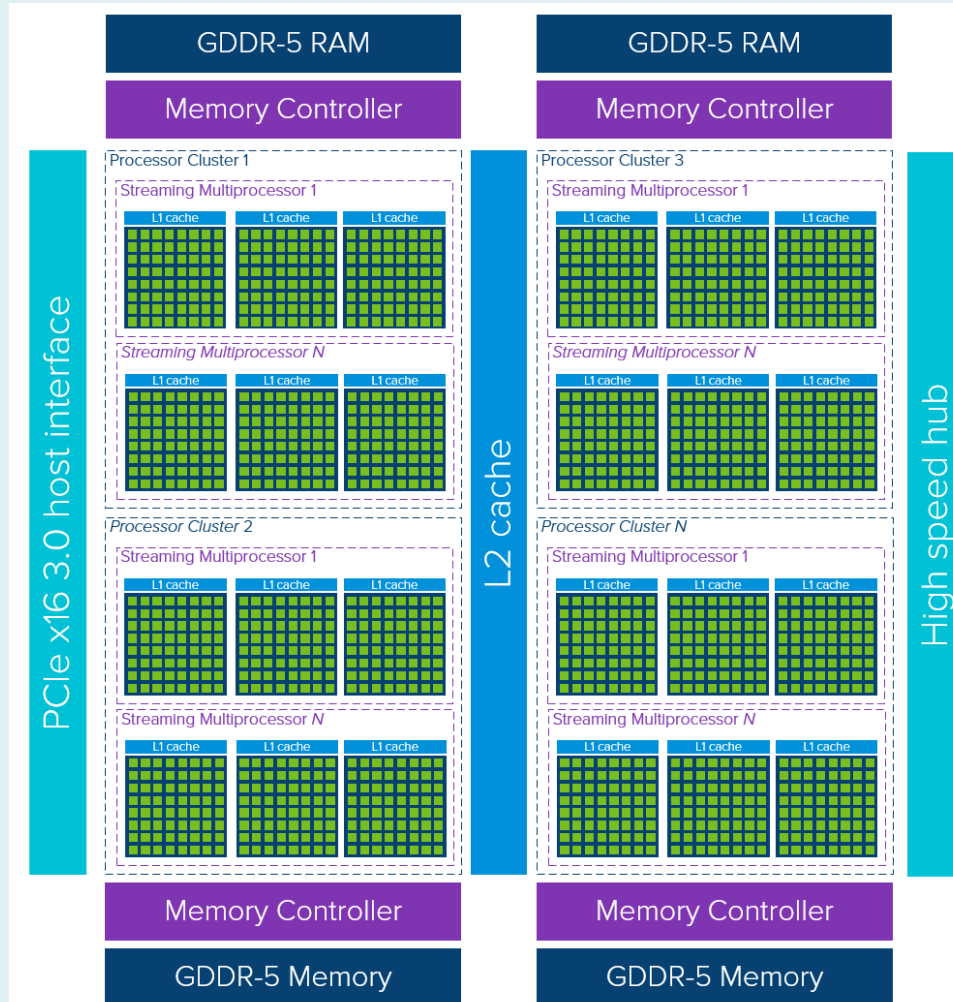
Different Memory Architectures - CPU



A high-level overview of modern CPU architectures indicates it is all about low latency memory access by using significant cache memory layers.

Different Memory Architectures - GPU

The nature of a GPU is all about putting available cores to work and it's less focused on low latency cache memory access.

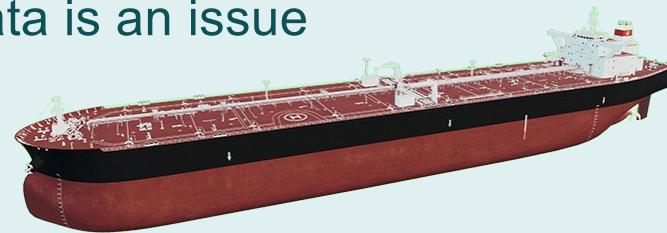


Different Memory Architectures – GPU vs GPU

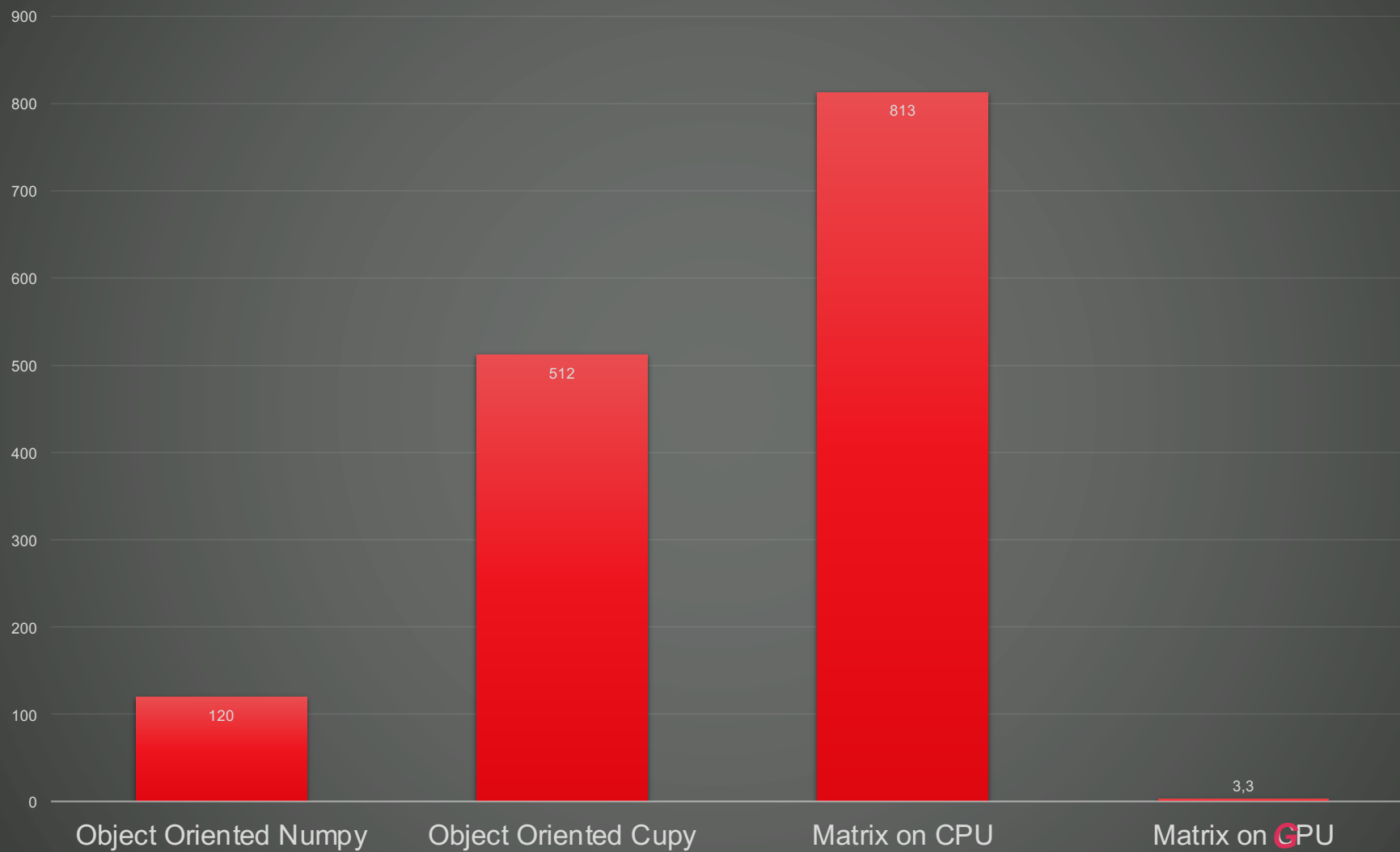
- CPU:
 - Moving small chunk of data & instructions is not an issue
 - Almost unlimited memory size (potential cache on SSD)



- GPU:
 - Moving in/out small chunk of data is an issue
 - Fixed memory size



EPMeans, k=7. In seconds



Different Purposes

Because GPUs can perform parallel operations on multiple sets of data, they are also commonly used for non-graphical tasks such as machine learning and scientific computation. Designed with thousands of processor cores running simultaneously, **GPUs enable massive parallelism** where each core is focused on making efficient calculations.

While GPUs can process data several orders of magnitude faster than a CPU due to massive parallelism, **GPUs are not as versatile as CPUs**. CPUs have large and broad instruction sets, managing every input and output of a computer, which a GPU cannot do

Most Common GPU / Neural Processing Unit

- NVIDIA GPU (Cuda Driver)
- Tensor Processing Unit (TPU) by Google
- Apple's Neural Engine (ANE)

Data Science on GPU/NPU

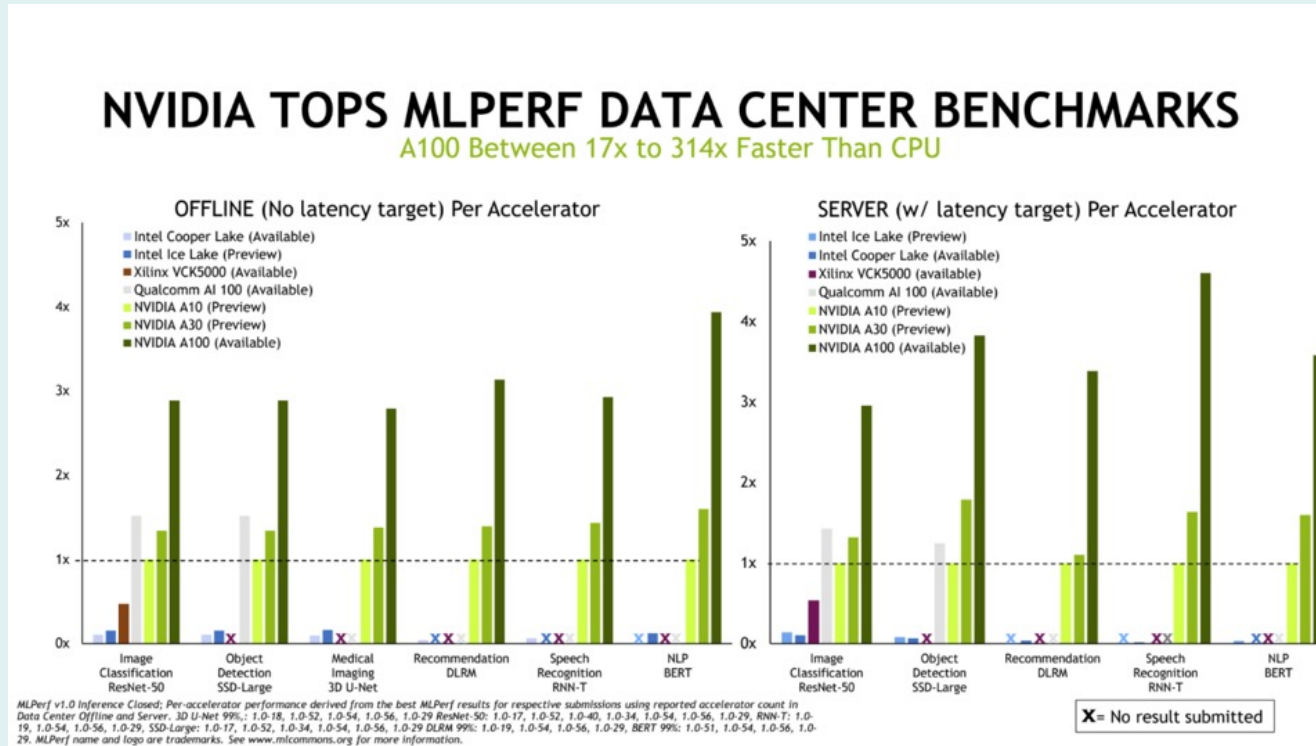
Why Data Science on GPU/NPU?

- DS/ML requires a lot of processing
 - Mostly based on mathematical transformations
 - Mostly based on matrices (NN, Tabular Data)
- GPU/ML are designed to efficiently perform:
 - Mathematical transformations in parallel
 - On matrices

Perfect fit!

Any benchmark?

NVIDIA TOPS MLPERF DATA CENTER BENCHMARKS A100 Between 17x to 314x Faster Than CPU



[NVIDIA Crushes Latest Artificial Intelligence Benchmarking Tests - Moor Insights & Strategy \(moorinsightsstrategy.com\)](https://moorinsightsstrategy.com)

GPU vs CPU – Conclusion

Conclusion

- GPU can be easy to very complex to use
- GPU might (dramatically) improve (or degrade) the performance
 - dedicated algorithms & code
 - in a specific environment, on the targeted data
 - with measures of performance & cost
- GPU are commonly used in ML/AI

**GPU can be
a game changer!**