Introduction to GPU Acceleration

View the Slides

https://github.com/ResearchComputing/Intro_GPU_Acceleration

Meet the User Support Team

Layla
Freeborn

Brandon
Reyes

Andy
Monaghan

Michael
Schneider

John
Reiland

Dylan
Gottlieb

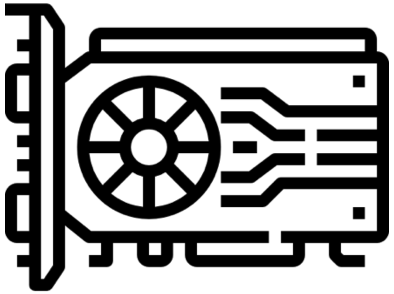
Mohal
Khandelwal

Ragan
Lee

Session Overview

GPU Icon

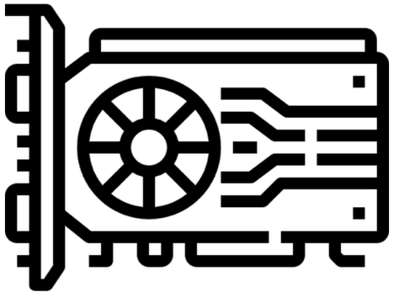
Session Overview



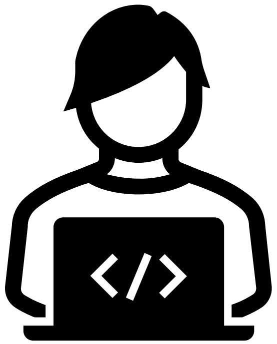
Basics Of GPUs

GPU Icon

Session Overview



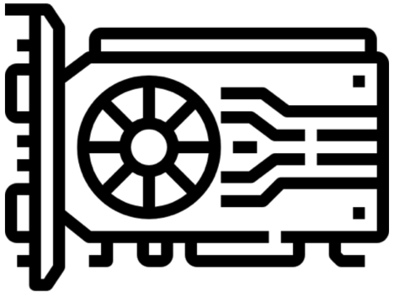
Basics Of GPUs



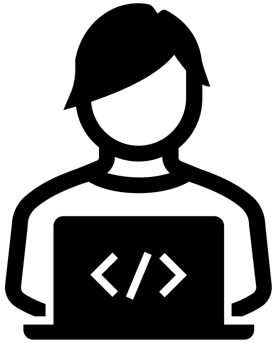
Code Optimization

GPU Icon

Session Overview



Basics Of GPUs



Code Optimization

Monitoring GPU Usage

GPU Icon

CPUs vs GPUs

Processing Unit

CPUs vs GPUs

Central

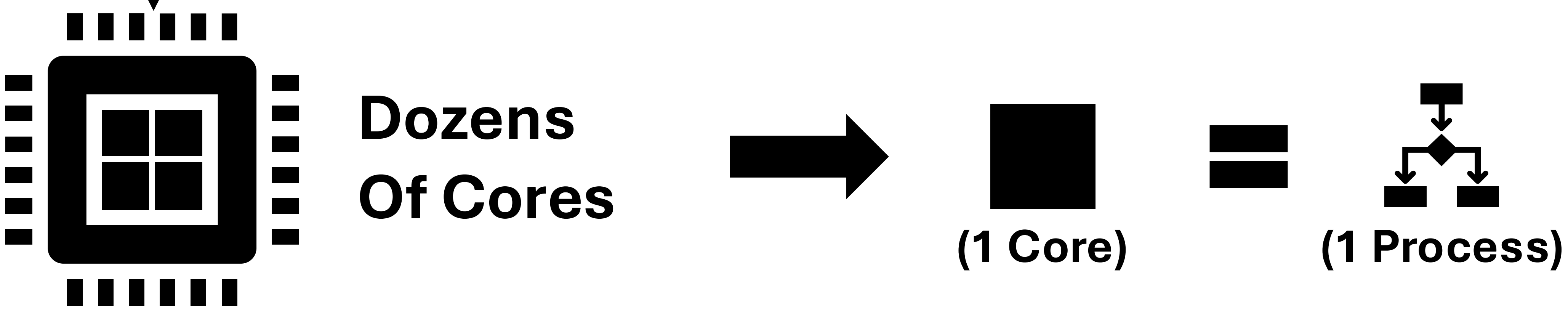
Processing Unit

CPUs vs GPUs

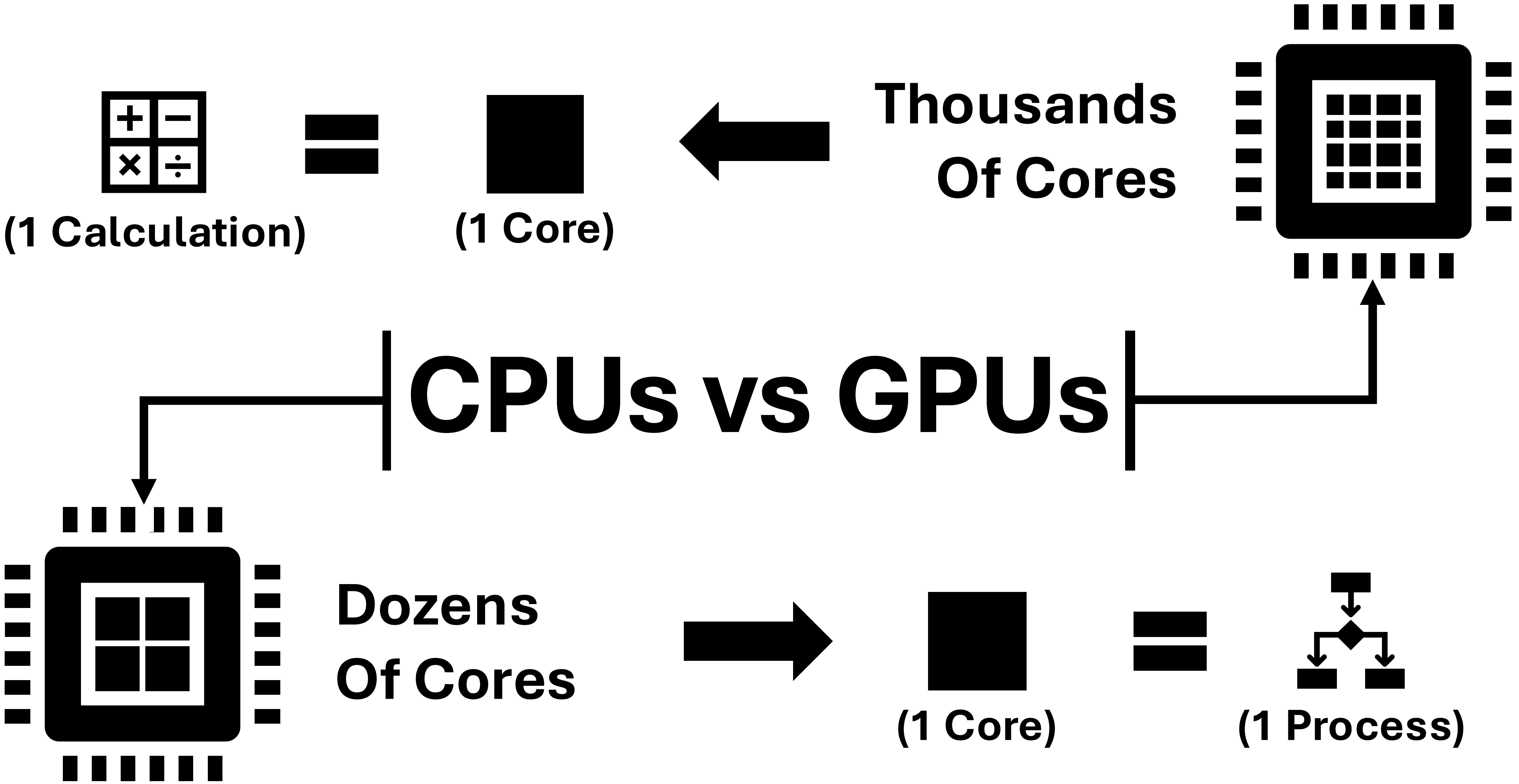
Central

Graphics

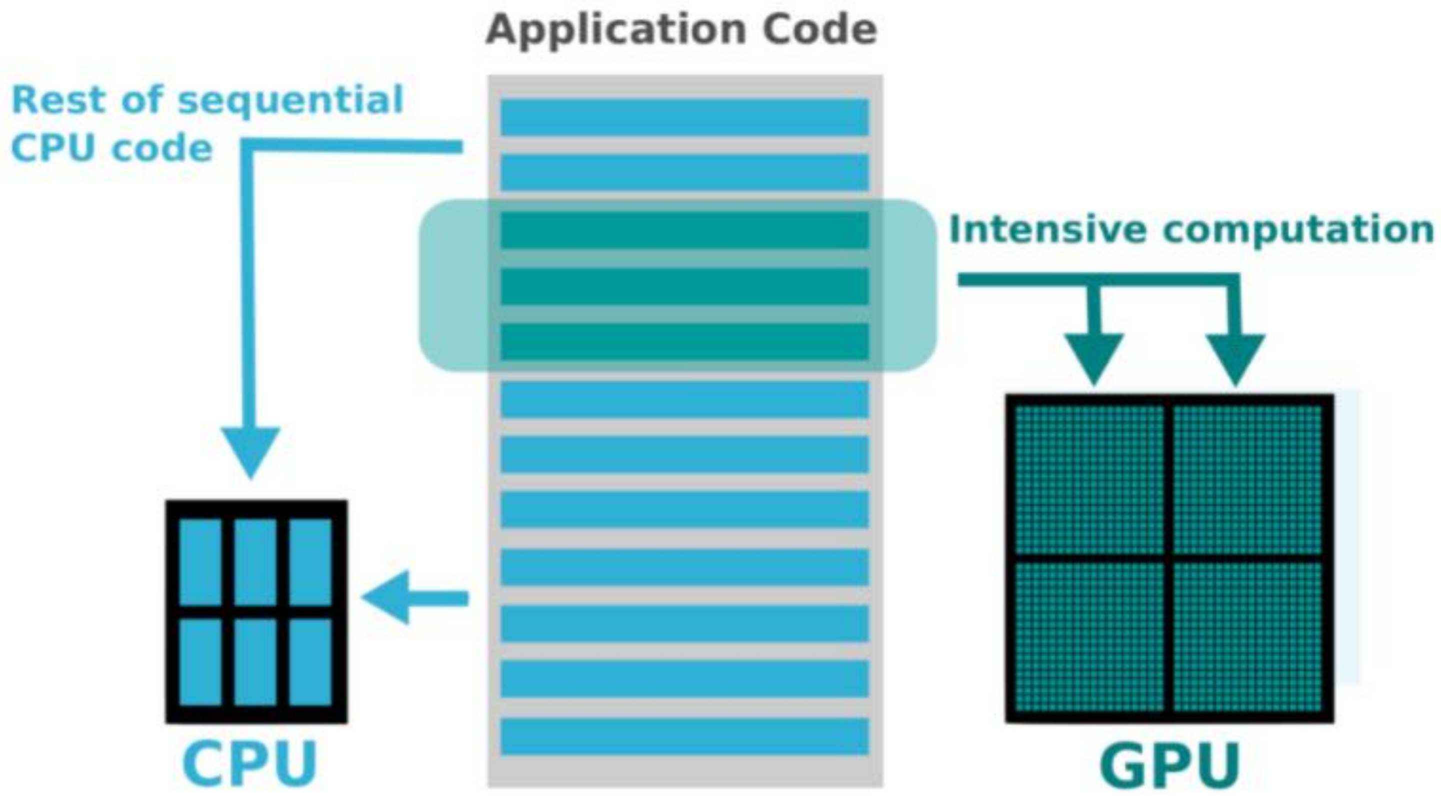
CPUs vs GPUs



CPUs vs GPUs


(1 Calculation)
(1 Core)
Thousands
Of Cores
Dozens
Of Cores
(1 Core)
(1 Process)

Computational Offloading



Graphic Source

GPU Acceleration Checklist

GPU Acceleration Checklist

☐ **Computational Intensity**

GPU Acceleration Checklist

- [ ] **Computational Intensity**
- [ ] **Algorithmic Complexity**

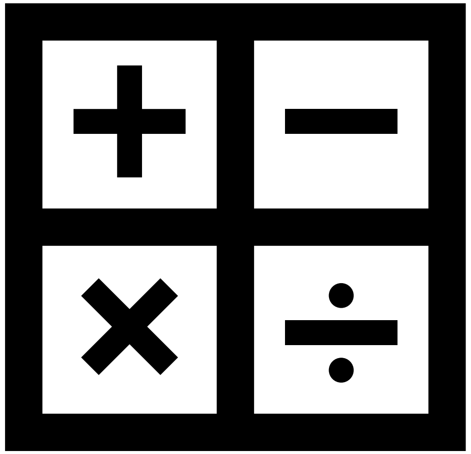
GPU Acceleration Checklist

- [ ] **Computational Intensity**
- [ ] **Algorithmic Complexity**
- [ ] **Data Type**

GPU Acceleration Checklist

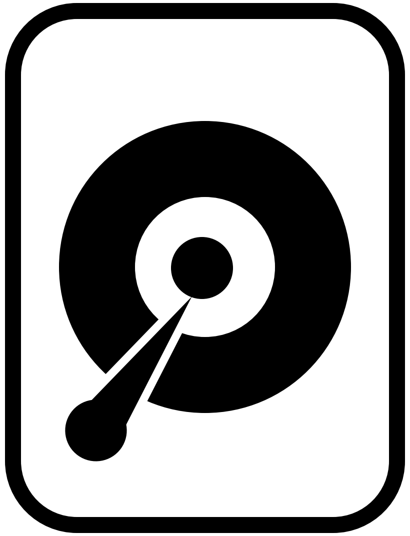
- ☐ **Computational Intensity**
- ☐ **Algorithmic Complexity**
- ☐ **Data Type**
- ☐ **Data Dependency**

Computational Intensity



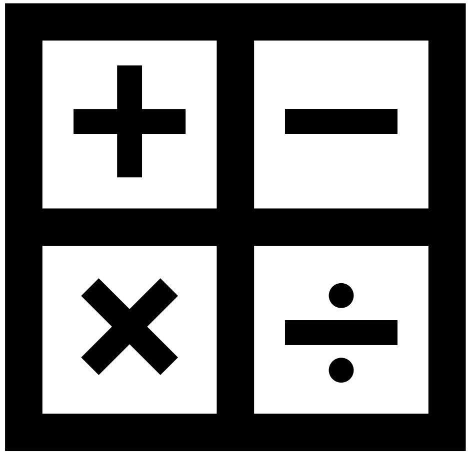
Calculations

VS



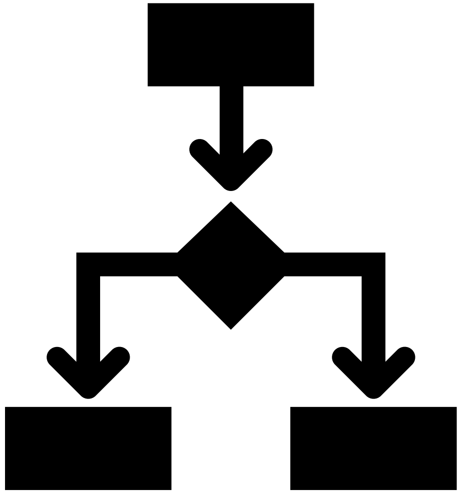
Data Access

Algorithmic Complexity



Calculations

VS



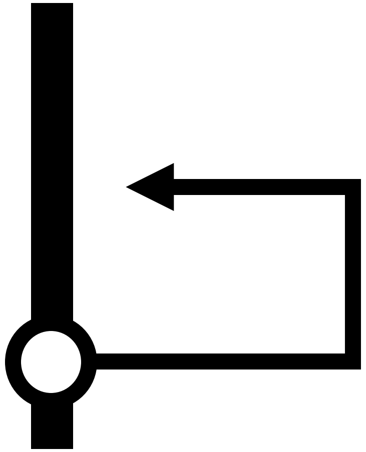
Branching Logic

Data Type

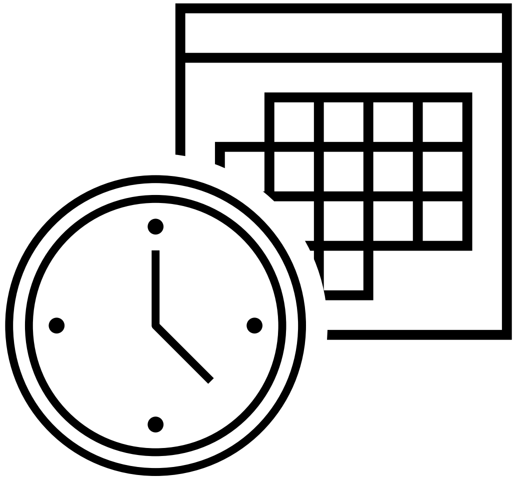
123 **vs** **“Text”**

Numeric **Complex Objects**

Data Dependency



Recursion



Temporal

Time / Dates

GPU Acceleration Checklist

- ☐ **Computational Intensity**
- ☐ **Algorithmic Complexity**
- ☐ **Data Type**
- ☐ **Data Dependency**

Alpine GPUs

	NVIDIA			AMD
Type	**A100**	**L40**	**GH200**	**MI100**
Cores	**7k**	**15K**	**17k**	**7.7k**
VRAM	**40 / 80**	**48**	**96**	**32**
Purpose	**General**	**Viz, AI Inference**	**AI Training, High Data I/O**	**Scientific**

Alpine GPUs

	NVIDIA			AMD
Type	**A100**	**L40**	**GH200**	**MI100**
Partition	**aa100**	**al40**	**gh200***	**ami100**
Nodes	**40 (8) / 80 (4)**	**3**	**2**	**2**
GPUs Per Node	**3**	**3**	**1**	**3**

Requesting GPUs

SLURM Directives:

--partition= <   >

--gres=gpu:<#>

--ntasks=<#>

Requesting GPUs

SLURM Directives:

--partition= <   >

--gres=gpu:<#>

--ntasks=<#>

sinteractive --partition=ami100 --gres=gpu:2 --ntasks=20

#SBATCH <directive>

Code Optimization

Libraries

Directives

Languages

GPU Icon

Key Terms

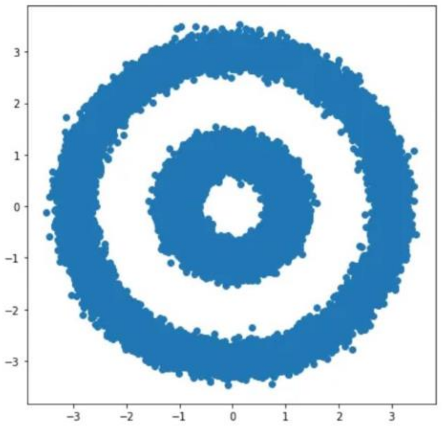
- Host == CPU
- Device == GPU
- Kernel == Functions launched on GPU

GPU Libraries – Drop in Replacement

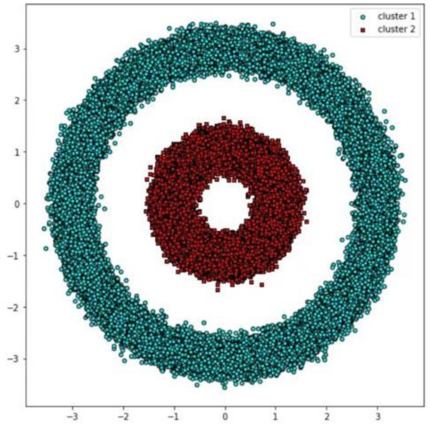
```
#create dataset with 100,000 points
from sklearn.datasets import make_circles
X, y = make_circles(n_samples=int(1e5), factor=.35, noice=.05)

#run DBSCAN clustering algorithm
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=0.6, min_samples=2)
y_db = db.fit_predict(X)
```

Dataset



Clusters



GPU Libraries – Drop in Replacement

```
#create dataset with 100,000 points
from sklearn.datasets import make_circles
X, y = make_circles(n_samples=int(1e5), factor=.35, noice=.05)
```

```
#run DBSCAN clustering algorithm
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=0.6, min_samples=2)
y_db = db.fit_predict(X)
```

GPU Libraries – Drop in Replacement

```
#create dataset with 100,000 points
from sklearn.datasets import make_circles
X, y = make_circles(n_samples=int(1e5), factor=.35, noice=.05)

#convert dataset to Pandas DataFrame




#run DBSCAN clustering algorithm
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=0.6, min_samples=2)
y_db = db.fit_predict(X)
```

GPU Libraries – Drop in Replacement

```
#create dataset with 100,000 points
from sklearn.datasets import make_circles
X, y = make_circles(n_samples=int(1e5), factor=.35, noice=.05)

#convert dataset to Pandas DataFrame
import pandas as pd
import cudf
X_df = pd.DataFrame({'fea%d'%i: X[:,i] for i in range(X.shape[1])})
X_gpu = cudf.DataFrame.from_pandas(X_df)

#run DBSCAN clustering algorithm
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=0.6, min_samples=2)
y_db = db.fit_predict(X)
```

GPU Libraries – Drop in Replacement

```
#create dataset with 100,000 points
from sklearn.datasets import make_circles
X, y = make_circles(n_samples=int(1e5), factor=.35, noice=.05)

#convert dataset to Pandas DataFrame
import pandas as pd
import cudf
X_df = pd.DataFrame({'fea%d'%i: X[:,i] for i in range(X.shape[1])})
X_gpu = cudf.DataFrame.from_pandas(X_df)

#run DBSCAN clustering algorithm
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=0.6, min_samples=2)
y_db = db.fit_predict(X)
```

❯

```
#run GPU-accelerated DBSCAN
from cuml import DBSCAN
```

GPU Libraries – Drop in Replacement

```
#create dataset with 100,000 points
from sklearn.datasets import make_circles
X, y = make_circles(n_samples=int(1e5), factor=.35, noice=.05)

#convert dataset to Pandas DataFrame
import pandas as pd
import cudf
X_df = pd.DataFrame({'fea%d'%i: X[:,i] for i in range(X.shape[1])})
X_gpu = cudf.DataFrame.from_pandas(X_df)
```

~~#run DBSCAN clustering algorithm~~
~~from sklearn.cluster import DBSCAN~~

❯

```
#run GPU-accelerated DBSCAN
from cuml import DBSCAN
```

```
db = DBSCAN(eps=0.6, min_samples=2)
y_db = db.fit_predict(X)
```

GPU Libraries – Drop in Replacement

```
#create dataset with 100,000 points
from sklearn.datasets import make_circles
X, y = make_circles(n_samples=int(1e5), factor=.35, noice=.05)
```

1

```
#convert dataset to Pandas DataFrame
import pandas as pd
import cudf
X_df = pd.DataFrame({'fea%d'%i: X[:,i] for i in range(X.shape[1])})
X_gpu = cudf.DataFrame.from_pandas(X_df)
```

2

```
#run GPU-accelerated DBSCAN
from cuml import DBSCAN
db = DBSCAN(eps=0.6, min_samples=2)
y_db = db.fit_predict(X)
```

GPU-Enabled Frameworks

TensorFlow

Keras

PyTorch

mxnet

GPU Compiler Directives

GPU Compiler Directives

OpenACC

Kernel directives
Generate parallel accelerator kernels
for the loop following the directive.

GPU Compiler Directives

OpenACC

Kernel directives
Generate parallel accelerator kernels for the loop following the directive.

```
//Hello_World_OpenACC.c
void Print_Hello_World()
{
  #pragma acc kernels
  for(int i=0; i<5; i++)
  {
    printf("Hello World!\n")
  }
}
```

GPU Compiler Directives

OpenACC

Kernel directives
Generate parallel accelerator kernels for the loop following the directive.

```
//Hello_World_OpenACC.c
void Print_Hello_World()
{
  #pragma acc kernels
  for(int i=0; i<5; i++)
  {
    printf("Hello World!\n")
  }
}
```

Data directives
Generate code to manage specific data operations to support parallelism

GPU Compiler Directives

Kernel directives

Generate parallel accelerator kernels for the loop following the directive.

```
//Hello_World_OpenACC.c
void Print_Hello_World()
{
  #pragma acc kernels
  for(int i=0; i<5; i++)
  {
    printf("Hello World!\n")
  }
}
```

Data directives

Generate code to manage specific data operations to support parallelism

```
//Hello_World_OpenACC.c
#pragma acc data copy(a)
{
  #pragma acc kernels
  for(int i=0; i<5; i++)
  {
    printf("Hello World!\n")
  }
}
```

Languages

- OpenCL (NVIDIA, AMD, & CPUs)
  - Flexible / portable option
- HIP (AMD -> NVIDIA)
  - AMD developed
  - Can convert CUDA code via `hippify`
- CUDA (NVIDIA only)
  - Most robust and largest developer community

Monitoring GPU Usage

- Nvidia-smi
- rocm-smi

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 510.47.03    Driver Version: 510.47.03    CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A100-PCI...  Off  | 00000000:21:00.0 Off |                    0 |
| N/A   36C    P0    40W / 250W |      0MiB / 40960MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA A100-PCI...  Off  | 00000000:81:00.0 Off |                    0 |
| N/A   36C    P0    40W / 250W |      0MiB / 40960MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   2  NVIDIA A100-PCI...  Off  | 00000000:E2:00.0 Off |                    0 |
| N/A   37C    P0    40W / 250W |      0MiB / 40960MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Troubleshooting GPU Workflows

- Is your application and/or code GPU accelerated?

  Confirm that you installed the GPU accelerated version!

- Does your application or code support **multi-**GPU acceleration?

- Is your application ROCM- or CUDA-aware?

  You can't run CUDA code on AMD GPUs. Not all applications are available for AMD GPUs.

- Can your application "see" the GPU?

- Did you request enough CPUs and RAM?

Documentation

https://curc.readthedocs.io/en/latest/

Survey and feedback

Survey: http://tinyurl.com/curc-survey18