

ACELERAÇÃO DE APLICAÇÃO DE IA COM NVIDIA RAPIDS



CONTEÚDO DO CURSO

- Introdução à GPU, CUDA e outros conceitos
- Vantagens no uso da GPU
- Introdução às soluções de aceleração da NVIDIA

RAPIDS

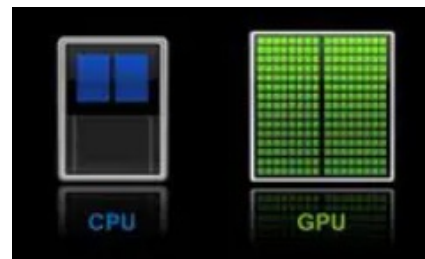
- cuDF – equivalente ao pandas
- cuML – equivalente ao sklearn
- Projeto completo abordando o ecossistema do RAPIDS
- Dask + cuDF + cuML

REQUISITOS

- Lógica de programação
- Programação básica em Python
- Aprendizagem de máquina

O QUE SÃO GPUS

- **GPUs** (*Graphics Processing Units* - Unidade de Processamento Gráfico) são processadores especializados projetados para acelerar o processamento gráfico e computacional.
- Diferentemente das CPUs, as GPUs possuem milhares de núcleos que podem executar tarefas simultaneamente, tornando-as altamente eficientes para tarefas paralelas.
- Foram originalmente desenvolvidas para renderizar gráficos em jogos e aplicações de computação gráfica, mas seu potencial se estendeu para outras áreas, como ciência de dados e aprendizado de máquina.
- Elas são altamente eficientes no processamento de grandes volumes de dados e cálculos complexos, graças à sua arquitetura paralela e capacidade de execução em massa.



Créditos da imagem: [NVIDIA](#)

GPUs EM APLICAÇÕES DE IA

- Com a evolução da tecnologia, as GPUs se tornaram ferramentas essenciais para acelerar algoritmos de aprendizado de máquina e realizar cálculos intensivos em diversas áreas, como pesquisa científica, análise de dados e simulações computacionais.
- As GPUs são amplamente utilizadas em plataformas de computação de alto desempenho, data centers e computação em nuvem, proporcionando um enorme impulso no processamento e desempenho de diversas aplicações.
- Portanto, a aceleração de GPU tem sido especialmente relevante para aplicações de inteligência artificial, nas quais algoritmos complexos e intensivos em dados são comuns.

GPUs vs. CPUs

- A CPU é composta por apenas alguns núcleos com muita memória de cache que pode lidar com alguns threads de software por vez.
- Por outro lado, uma GPU é composta por centenas de núcleos que podem lidar com milhares de threads simultaneamente, ou seja, milhares de operações ao mesmo tempo.
- Entretanto, as CPUs certamente permanecem essenciais. Enquanto CPUs permanecem ideais para processamento em série, GPUs são ideais para processamento paralelo.

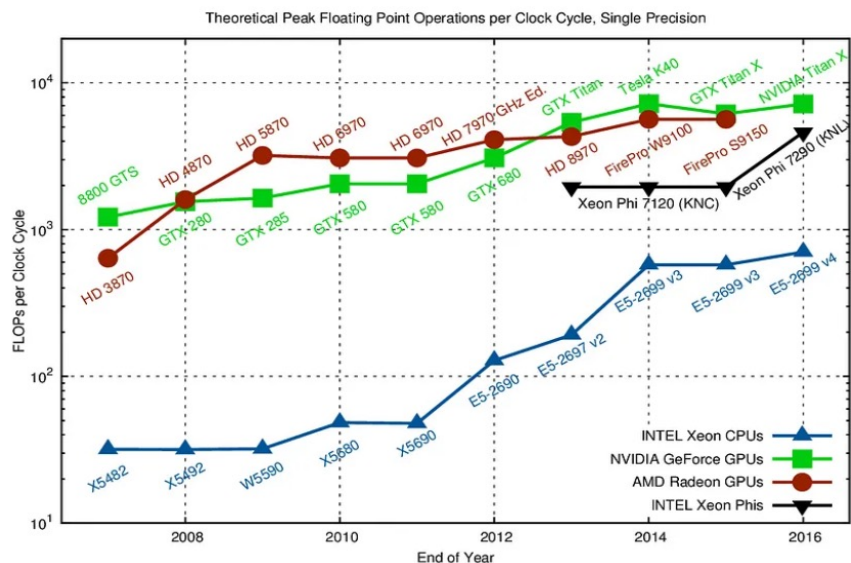
Para aplicações de IA:

- GPU: acelera o processamento executando simultaneamente várias tarefas em milhares de núcleos, proporcionando velocidade e eficiência para algoritmos intensivos em dados.
- CPU: Embora seja projetada para tarefas em série, a CPU também pode executar aplicações de IA, mas em uma escala menor. Ela é mais adequada para tarefas sequenciais e complexas.

GPUs vs. CPUs

- A GPU é um processador de chip único usado para extensas computações gráficas e matemáticas que liberam ciclos de CPU para outros trabalhos.
- A principal diferença entre GPUs e CPUs é que as GPUs dedicam proporcionalmente mais transistores para unidades lógicas aritméticas e menos para caches e controle de fluxo em comparação com as CPUs.
- Comparado à CPU, tende a ter mais núcleos lógicos (unidades lógicas aritméticas ou ALUs, unidades de controle e cache de memória).

As GPUs (vermelho/verde) teoricamente podem realizar de 10 a 15 vezes mais operações que as CPUs (em azul)



Créditos: fast.ai

VANTAGENS DA GPU PARA APLICAÇÕES DE IA

- A principal vantagem quanto às GPUs é o **paralelismo**: elas possibilitam a execução paralela de tarefas, permitindo processar múltiplos cálculos simultaneamente.
- Com GPUs, é possível acelerar o treinamento e a inferência de modelos de aprendizado de máquina e redes neurais.
- A capacidade de executar tarefas em paralelo nas GPUs reduz o tempo necessário para treinar modelos e realizar a análises dos resultados.

VANTAGENS DA GPU PARA APLICAÇÕES DE IA

- GPUs oferecem melhor desempenho para algoritmos de processamento de imagem e visão computacional em tempo real.
- Com a crescente demanda por análises em tempo real, as GPUs proporcionam a velocidade necessária para responder rapidamente a eventos e tomadas de decisão em tempo hábil.
- Como a aceleração de GPU permite lidar com conjuntos de dados cada vez maiores e complexos, a pesquisa científica e a descoberta de insights em diversas áreas de conhecimento é impulsionada.

CONDIÇÕES PARA USAR GPU EM SEU PROJETO

- Se o sistema e a GPU são compatíveis
- Se a linguagem e as bibliotecas possuem suporte à GPU
- Como configurar o código do projeto para utilizar a GPU
- Também é importante fazer testes e avaliações do desempenho do código, para ver o quão viável é a transferência para GPU.

GPUs

Algumas ferramentas que oferecem suporte específico à GPUs:

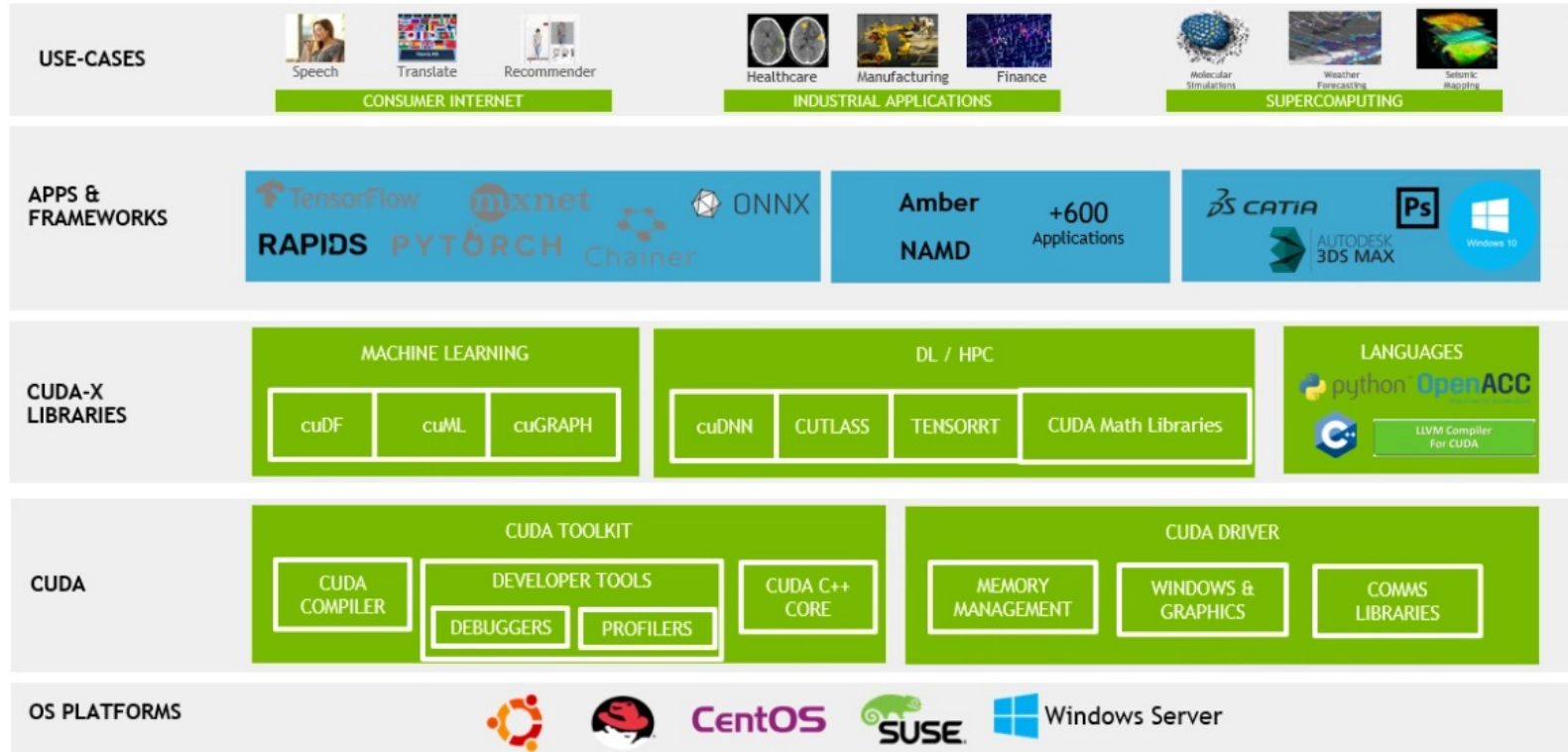
- APIs para acesso direto à GPU – exemplo: CUDA e OpenCL
- Implementações paralelas de algoritmos populares, como convoluções e operações matriciais.
- Funções para mover dados da CPU para GPU – e vice-versa

CUDA

- **CUDA** (*Compute Unified Device Architecture*) é uma plataforma de computação paralela desenvolvida pela NVIDIA para programação em GPUs.
- Permite desenvolver programas em linguagens como C/C++ (e inclusive linguagens ainda mais antigas, como Fortran), podendo ser executadas em GPUs da NVIDIA.
- Foi projetada para ser preparada para a execução de tarefas que exigem alto desempenho computacional.
- Através da CUDA, possuímos acesso a um kit de ferramentas de desenvolvimento com compiladores e depuradores para auxiliar na programação com GPU.



CUDA



Créditos da imagem: NVIDIA

SOLUÇÕES DA NVIDIA PARA ACELERAÇÃO

- A NVIDIA oferece atualmente algumas soluções que proporcionam aceleração significativa para processamento de dados e análise de vídeo.
- Essas soluções proporcionam um aumento significativo de desempenho e eficiência, abrindo possibilidades para aplicações de alto impacto em diversos setores.

RAPIDS

- Oferece um amplo ecossistema de bibliotecas de GPU para acelerar projetos de Ciência de Dados e Machine Learning.
- Possibilita utilizar as GPUs para acelerar tarefas comuns, como carregamento, manipulação, análise e visualização de dados.

DeepStream

- Plataforma de análise de vídeo em tempo real.
- Projetado com foco em processamento de streams (transmissões) de vídeo, permitindo acelerar técnicas avançadas de visão computacional.

RAPIDS - NVIDIA

The RAPIDS logo features the word "RAPIDS" in a bold, white, sans-serif font. It is centered within a rectangular area that has a purple-to-blue gradient. Overlaid on this gradient are several semi-transparent, overlapping geometric shapes, including triangles and parallelograms, in various shades of purple and blue, creating a dynamic, layered effect.

RAPIDS

O QUE É O RAPIDS

- O RAPIDS é um conjunto de bibliotecas que permite a execução de ponta a ponta de pipelines de ciência de dados, machine learning e inteligência artificial por meio de GPUs.
- Fornece velocidade inigualável com APIs familiares que correspondem às bibliotecas mais populares de Ciência de Dados e Machine Learning para Python.
- A partir do uso dessas bibliotecas, exclui-se a necessidade de saber os detalhes técnicos para aprender a lidar com paralelismo e linguagens específicas para GPU, como o OpenCL ou CUDA.
- Excelente para fazer o processamento de dados de forma acelerada e distribuída em GPUs através de uma interface amigável em Python.

RAPIDS

RAPIDS: Libraries for End to End GPU Data Science
<https://rapids.ai>

VANTAGENS DO RAPIDS

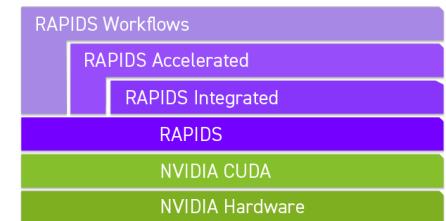
- **Integração facilitada e familiar** – facilita a implementação pois requer alterações mínimas no código, já que utiliza a mesma lógica das bibliotecas mais conhecidas de inteligência artificial. Portanto, não exige aprender novas ferramentas complicadas ou novas sintaxes.
- **Aumento na produtividade** – possibilita reduzir o tempo gasto para treinamento e outros processamentos, exigindo assim um tempo de espera muito menor para as fases de verificação, treinamento e validação.
- **Aumento na velocidade e na acurácia** - ao permitir um deploy mais rápido permite explorar e testar mais os dados, permitindo uma análise mais profunda e consequentemente as chances de aumentar a acurácia dos modelos.
- **Redução de custo** – permite-se aumentar a eficiência do data center. Além disso, possui uma alta escalabilidade.
- **Open source** - software de código aberto personalizável com suporte da NVIDIA e desenvolvido no Apache Arrow.

VANTAGENS DO RAPIDS

- Em muitas situações, a complexidade e o custo de se trabalhar com uma GPU está em tirar os dados da memória do host e trazê-lo para a memória da GPU
- O RAPIDS faz esse processo de forma otimizada, deixando os dados no cache do GPU, desse modo aumentando a performance
- Quanto mais conseguirmos processar dentro da GPU, mais aceleração teremos.
- Com o uso do RAPIDS, calcula-se um potencial de 100x até 200x mais performance (embora esse fator de aceleração varie bastante entre os diferentes modelos de GPUs).

ECOSSISTEMA DE CIÊNCIA DE DADOS E MACHINE LEARNING

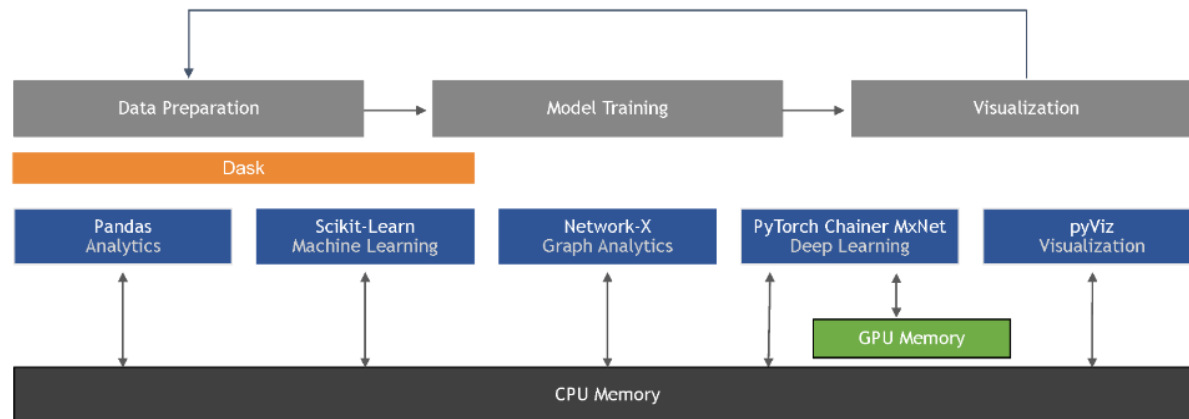
- O ecossistema de Data Science e Machine Learning atualmente possui várias bibliotecas populares e importantes (como Pandas, Scikit-learn)
- São bibliotecas excelentes e considerados por muitos como pilares fundamentais para a comunidade; porém, foram desenvolvidos para trabalhar na memória de CPUs.
- O RAPIDS surge como uma solução para implementar as funções dessas bibliotecas, mas processando na GPU. Para isso, oferece um ecossistema onde é disponibilizado diversas bibliotecas, que são funcionalmente equivalentes. Inclusive há alta compatibilidade semântica, onde muitas vezes é necessário apenas trocar o import.
- O código que funciona no Pandas pode funcionar na biblioteca correspondente do RAPIDS (nesse exemplo, o cuDF).



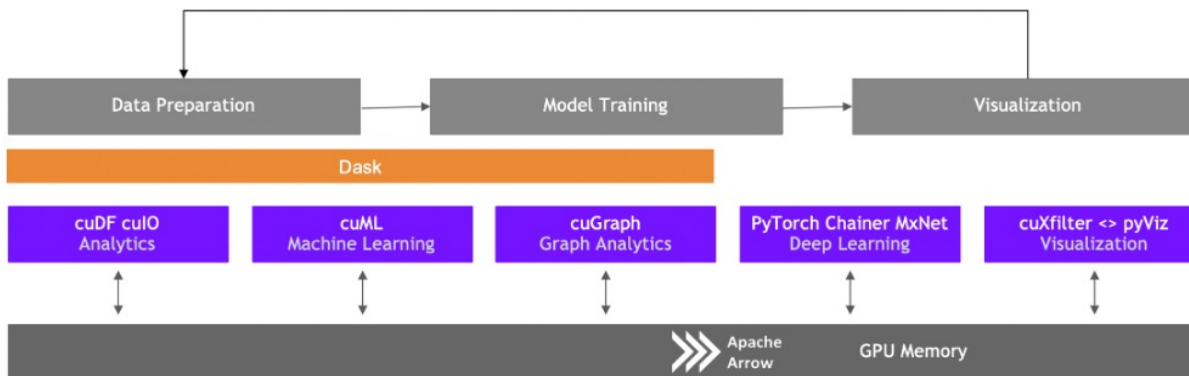
Créditos: <https://rapids.ai>

ECOSSISTEMA TRADICIONAL VS. ECOSSISTEMA RAPIDS

Ecosystema
tradicional de
Ciência de Dados



Ecosystema
RAPIDS



Créditos das imagens: <https://rapids.ai/>

APACHE ARROW

- O RAPIDS trabalha conectada com o **Apache Arrow**, dentro da memória GPU.
- O Apache Arrow fornece um formato de memória colunar eficiente e interoperável. Transforma uma tabela em um dado colunar.
- Na prática, deixa o dado mais rápido de ser acessado.
- Melhora-se o desempenho e a eficiência das operações de leitura, gravação e manipulação de dados.
- O Apache Arrow contribui para um processamento acelerado de dados em GPU de maneira eficiente e escalável, reduzindo a sobrecarga de conversão de dados e melhorando a eficiência.



<https://arrow.apache.org>

BIBLIOTECAS/APIS DO RAPIDS

O RAPIDS é capaz de reimplementar famosas bibliotecas distribuídas em Python para trabalhar de forma eficiente com as GPUs, oferecendo mais eficiência e velocidade do que seria possível com uma CPU:

- **cuDF** – biblioteca de análise de dados distribuída que usa a plataforma CUDA, para executar operações de processamento de dados. Foi projetada para ser uma alternativa ao **Pandas**.
- **cuML** – uma biblioteca de ML que usa a CUDA para executar algoritmos de ML em GPUs. É uma alternativa ao **scikit-learn**.
- **cuGraph** – biblioteca de análises de grafos. É uma alternativa ao **NetworkX**
- **cuSpatial** – bibliotecas espaciais alternativas ao **GeoPandas** e **Shapely**
- **cuSignal** – alternativa ao **scipy.signal**
- **cuCIM** – alternativa ao **scikit-image**.

Mais projetos: <https://github.com/rapidsai> (que ainda podem ser lançados em um futuro próximo ou distante)

BENEFÍCIOS DO RAPIDS NA PRÁTICA

- O RAPIDS facilita o processo de transferir o processamento da CPU para GPU.
- Muitas vezes é necessário apenas trocar o import de uma biblioteca para passar a usar a GPU.
- Roda em hardwares baseados em CUDA.
- Funciona em múltiplas plataformas, como PCs, Data Centers e na Nuvem.
- É possível instalar e utilizar em seu ambiente de desenvolvimento, desde que sua máquina possua uma GPU disponível e compatível.

FERRAMENTAS DE INTEGRAÇÃO COM RAPIDS

Pode-se integrar com:

- Dask – usado para dimensionar a carga de trabalho para diversas GPUs.
- Apache Spark – passou a oferecer aceleração de GPU totalmente integrada através do Spark Rapids, para acelerar os fluxos de trabalho sem alteração de código, aumentando o desempenho e reduzindo custos.
- Dask SQL – integrado com o Rapids SQL, lib open source que traz funcionalidades de SQL ao Dask; permite escrever consultas SQL para trabalhar com grandes conjuntos de dados distribuídos, escalando facilmente para grandes volumes de dados
- XGBoost – biblioteca open source para algoritmos de ML baseado em árvores de decisão.

• *Ecossistema completo: consultar documentação oficial*



Atualmente, RAPIDS possui integração para centenas de serviços diferentes, que vão desde open source até softwares comerciais. A lista cresce a cada ano, podendo ser integrações oficiais (da equipe do RAPIDS) ou da comunidade

CASOS DE USO EM EMPRESAS E CORPORAÇÕES

Imagine agora como empresas grandes e que lidam com uma quantidade imensa de dados que devem ser analisados podem usufruir do RAPIDS.

Algumas das empresas que utilizam o RAPIDS: Amazon, Walmart, AT&T e Bumble.

Walmart

– utilizado para aprimorar a previsão de demanda



Créditos da imagem: Sundry Photography /Shutterstock

AT&T

– potencializou diversas operações de ciência de dados



Amazon

– uso de Graph Neural Networks (GNN) para diversas aplicações, como sistemas de recomendação, detecção de fraudes e segurança cibernética



A Amazon e a NVIDIA firmaram uma parceria no desenvolvimento da Enterprise Deep Graph Library (DGL) para GNNs de grande escala

IMPLEMENTAÇÕES

- Linguagens de programação: Python (mais popular atualmente) e C++
- Bibliotecas Python – TensorFlow, PyTorch
- Bibliotecas C++ – API CUDA, ArrayFire, Boost.Compute

REQUISITOS PARA USO

- GPU NVIDIA Pascal – ou superior com capacidade computacional acima de 6.0
- Sistema Operacional Ubuntu 20.04 ou 22.04; CentOS 7; Rocky Linux 8; ou WLS2 no Windows;
- Versão recente do CUDA & Drivers NVIDIA correspondentes

- ✓ [CUDA 11.2](#) with Driver 460.27.03 or newer
- ✓ [CUDA 11.4](#) with Driver 470.42.01 or newer
- ✓ [CUDA 11.5](#) with Driver 495.29.05 or newer
- ✓ [CUDA 11.8](#) with Driver 520.61.05 or newer
- ✓ [CUDA 12.0](#) with Driver 525.60.13 or newer for pip installations only

- <https://docs.rapids.ai/install>

INSTALAÇÃO

O RAPIDS oferece diferentes opções para fazer a implementação:

- Conda
 - Docker
 - WLS2
 - PIP (mais recente)
 - GitHub (build da fonte)
-
- Dica: Instalação através do container NGC Nvidia
<https://catalog.ngc.nvidia.com/orgs/nvidia/containers/tensorflow>

INSTALAÇÃO NO COLAB

- Atualmente, no Colab temos uma forma prática e rápida que é através de um repositório <https://github.com/rapidsai/rapidsai-csp-utils>
- Esse repositório contém um script (install_rapids.py) que vai instalar corretamente o RAPIDS de modo compatível com a GPU do Colab e sua respectiva versão CUDA.
- Desse modo, podemos instalar todos os pacotes necessários no Colab usando apenas dois comandos ao todo.

```
!git clone https://github.com/rapidsai/rapidsai-csp-utils.git  
!python rapidsai-csp-utils/colab/pip-install.py
```

INSTALAÇÃO EM AMBIENTE LOCAL

Um modo bastante prático de instalar o RAPIDS em ambiente local é usando o NGC - Repositório de modelos de redes neurais e de containers da NVIDIA.

Para rodar em sua máquina, basta seguir os passos

<https://catalog.ngc.nvidia.com/orgs/nvidia/teams/rapidsai/containers/rapidsai>

- Base e Runtime - esse último vem junto com alguns exemplos de código dentro
- Pré-requisitos: Docker CE, Drive de CUDA (da sua GPU), plugin da NVIDIA do docker (para o docker conseguir acessar a GPU); e então, execute os comandos listados.
- Obs: Ao instalar o driver da NVIDIA da placa de vídeo, tem que aparecer detalhes sobre sua GPU ao executar o comando: `!nvidia-smi` assim que sabemos se o driver foi instalado corretamente.

PROBLEMAS QUE PODEM ACONTECER NA INSTALAÇÃO

Devido às constantes atualizações e diferentes combinações de ambientes e versões, recomenda-se consultar na página oficial da NVIDIA as soluções para os problemas que podem ocorrer. Essa página contém informações atualizadas referente ao método de instalação escolhido, portanto sugere-se checar se bate todos os pré-requisitos ou avisos referentes ao seu sistema operacional e ambiente

- <https://docs.rapids.ai/install#troubleshooting>

CONTEÚDO

Bibliotecas que podem ser facilmente integradas com pipelines comuns de ciência de dados.

- **CuDF**
- **CuML**
- **CuPy**

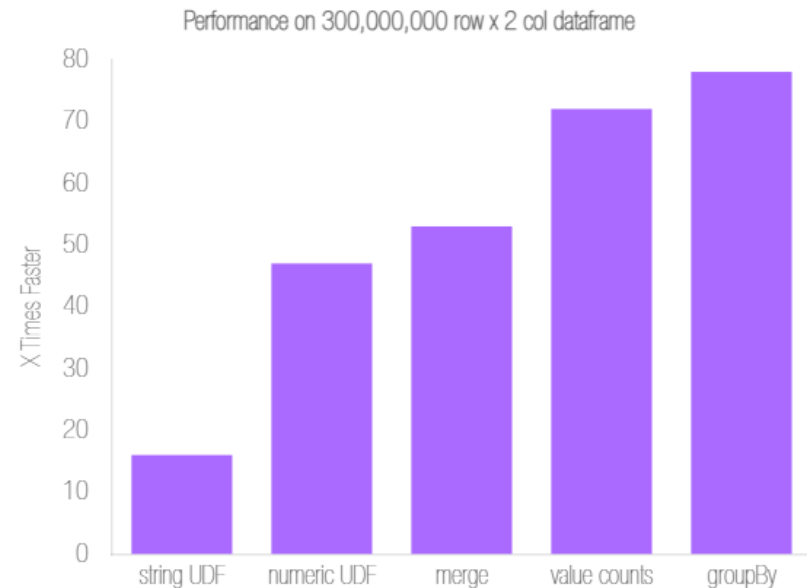
- O **cuDF** é um pacote dentro do ecossistema RAPIDS que permite migrar facilmente seus fluxos de trabalho com Pandas da CPU para a GPU.
- Pode ser utilizado para carregar, unir, agregar, filtrar e manipular grandes conjuntos de dados, aproveitando modelos de programação de GPU.
- Os cálculos podem aproveitar a imensa paralelização fornecida pelas GPUs.
- Por fornecer uma API semelhante ao Pandas, os desenvolvedores ou cientistas de dados não precisam se aprofundar no modelo de programação CUDA.

cuDF - COMPARATIVO

- As operações são quase as mesmas dos pandas e podem facilmente substituir as operações do Pandas em nosso pipeline tradicional de ciência de dados.
- Embora os resultados possam variar um pouco em diferentes GPUs, deve ficar claro que a aceleração da GPU pode fazer uma diferença significativa.
- Podemos obter resultados muito mais rápidos com o mesmo código.

Configurações comparadas no teste:

- AMD EPYC 7642 (usando 1x 2.3GHz CPU core) com 512GB;
- NVIDIA A100 80GB (1x GPU) com pandas v1.5 e cuDF v23.02.



Créditos: rapids.ai

cuDF - COMPARATIVO

- Um outro comparativo pode ser observado no gráfico ao lado, realizado no dataset California road network (*Leskovec 2009*)
- Obs: o gráfico está em escala logarítmica, portanto, não-linear



Operação	cuDF (s)	Pandas DF (s)	Aceleração
Read CSV	7.532238	67.287993	8.9x
Reverse DF	0.031103	1.622508	52.2x
Merge DFs	2.354040	80.349599	34.1x
Drop column and rows	4.165711	218.142479	52.4x
Concat DFs	0.345340	2.469050	7.1x

Créditos: [Ahmedur Shovon](#)

PANDAS vs. cuDF

Criar um DataFrame e somar os valores de uma determinada coluna

Com o Pandas

```
import pandas as pd
```

```
df = pd.DataFrame()  
df['id'] = [0, 1, 2, 2, 3, 3, 3]  
df['val'] = [float(i + 10) for i in range(7)]  
print(df)
```

```
soma = df['val'].sum()  
print(soma)
```

91.0

Com o cuDF

```
import cudf
```

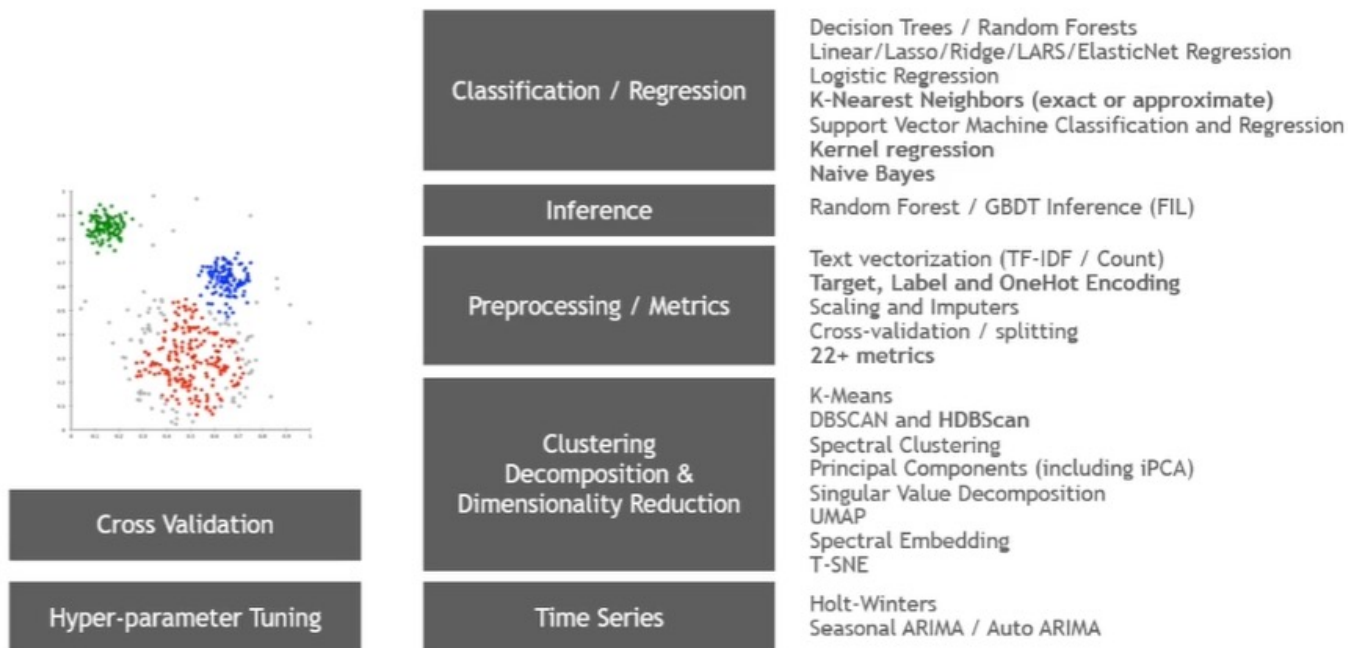
```
df = cudf.DataFrame()  
df['id'] = [0, 1, 2, 2, 3, 3, 3]  
df['val'] = [float(i + 10) for i in range(7)]  
print(df)
```

```
soma = df['val'].sum()  
print(soma)
```

91.0

- O **cuML** é um pacote dentro do ecossistema RAPIDS que permite utilizar funções do Scikit-Learn na GPU, com uma API que procura ser o mais similar possível.
- Suporta uma ampla gama de algoritmos, como regressão, classificação e clustering.
- Aproveita o poder de processamento paralelo das GPUs para um desempenho rápido com alta escalabilidade.
- Reduz o tempo necessário para desenvolver e implementar modelos de Machine Learning.
- Aumentando a velocidade de treinamento aumenta-se a produtividade dos cientistas de dados e engenheiros de ML.

- Algumas das abordagens suportadas pelo cuML



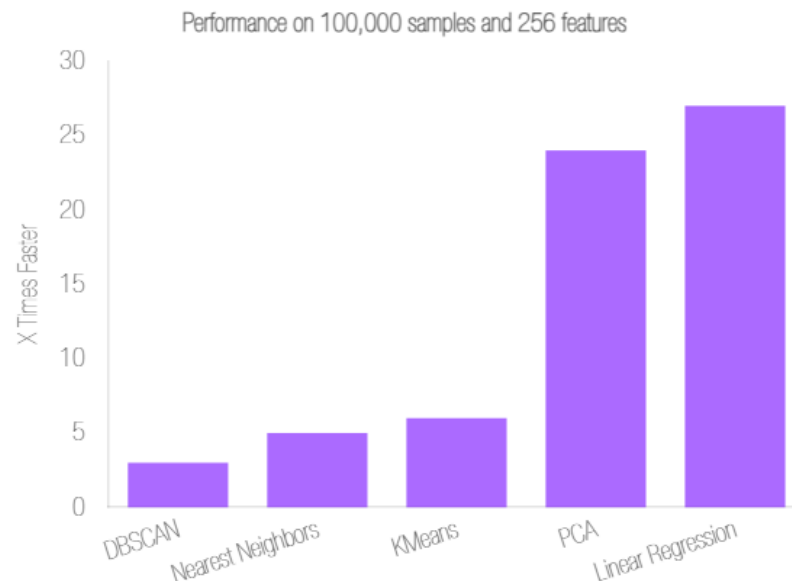
- Obs: é uma lista que pode facilmente ficar desatualizada porque o RAPIDS é atualizado com frequência e em várias versões pode ser incluído o suporte a novos métodos.

cuML - COMPARATIVO

- cuML traz enormes acelerações para modelagem de ML com uma API que corresponde ao Scikit-learn.
- Observa-se uma considerável aceleração pois as operações são migradas da CPU para a GPU.
- Assim como acontece para o caso do cuDF, com o cuML podemos obter resultados muito mais rápido e usando um código familiar.

Configurações comparadas no teste:

- AMD EPYC 7642 (usando 1x 2.3GHz CPU core) com 512GB;
- NVIDIA A100 80GB (1x GPU) com scikit-learn v1.2 e cuML v23.02.



Créditos: rapids.ai

SCIKIT-LEARN vs. cuML

Implementando uma Regressão Linear para dados dados empíricos

Com o Scikit-learn

```
import sklearn
from sklearn.linear_model import LinearRegression

linear_regression = LinearRegression()

linear_regression.fit(np.expand_dims(x, 1), y_noisy)

inputs = np.linspace(start=-5, stop=5, num=1000000)

outputs = linear_regression.predict(np.expand_dims(inputs, 1))
```

Com o cuML

```
import cuml
from cuml.linear_model import LinearRegression as LinearRegressionGPU

linear_regression_gpu = LinearRegressionGPU()

linear_regression_gpu.fit(cp.expand_dims(cp.array(df['x']), 1), y_noisy)

inputs = np.linspace(start=-5, stop=5, num=1000000)
df_cudf = cudf.DataFrame({'inputs': inputs})

outputs_gpu = linear_regression_gpu.predict(df_cudf[['inputs']])
```


RAPIDS E DASK

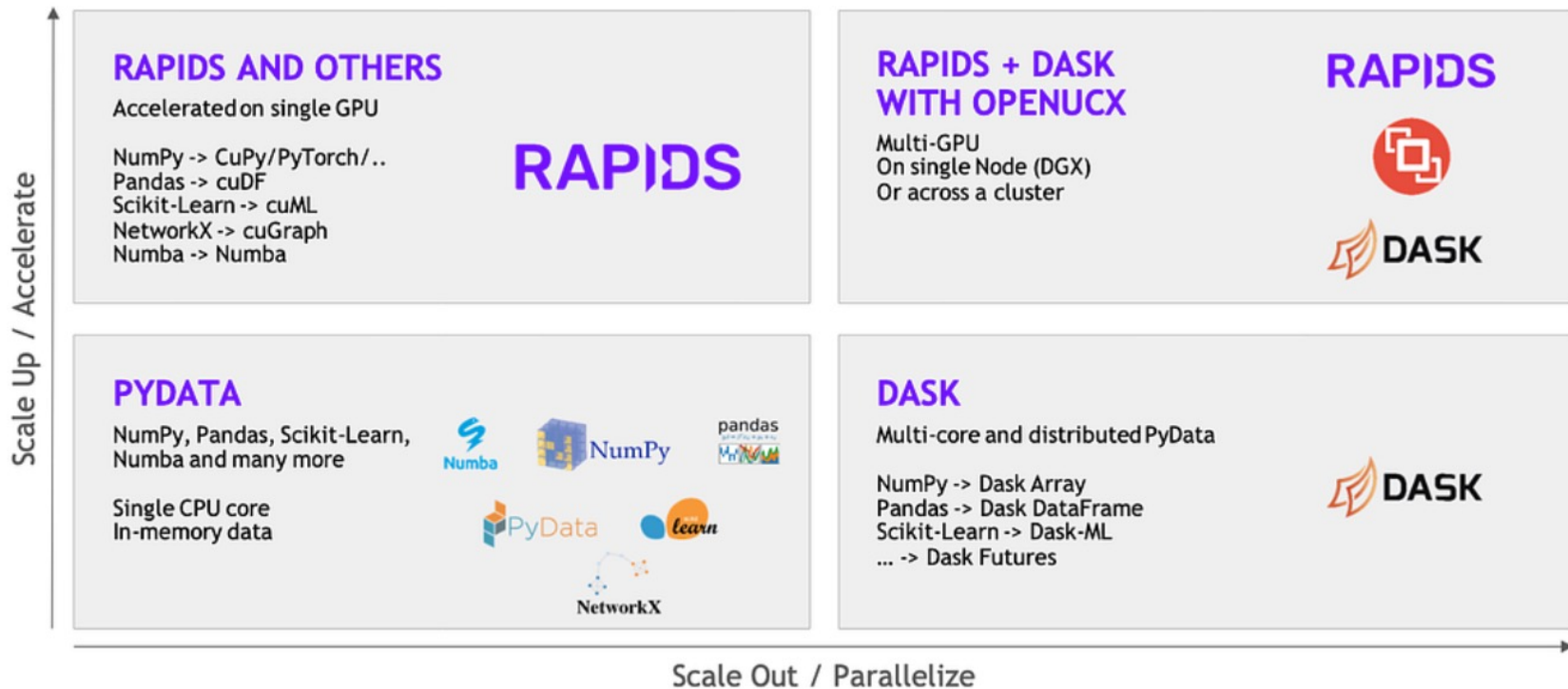
- **Dask** é uma biblioteca flexível para computação paralela em Python que torna o dimensionamento do seu fluxo de trabalho suave e simples.
- É utilizada para processamento distribuído, que se integra com o RAPIDS
- Na CPU, o Dask usa Pandas (NumPy) para executar operações em paralelo em partições DataFrame (array).
- Não é necessário saber sobre paralelismo para implementar, pois a biblioteca abstrai todos os processos complexos.
- O algoritmo se encarrega de fazer a distribuição, sincronização e gerenciamento.



RAPIDS E DASK – QUANDO USAR

- Se você deseja distribuir seu fluxo de trabalho em várias GPUs.
- Possui mais dados do que pode caber na memória em uma única GPU; por exemplo, em casos onde o dataset é maior que a memória disponível da GPU.
- Quando deseja-se analisar dados espalhados por vários arquivos de uma só vez.

RAPIDS E DASK



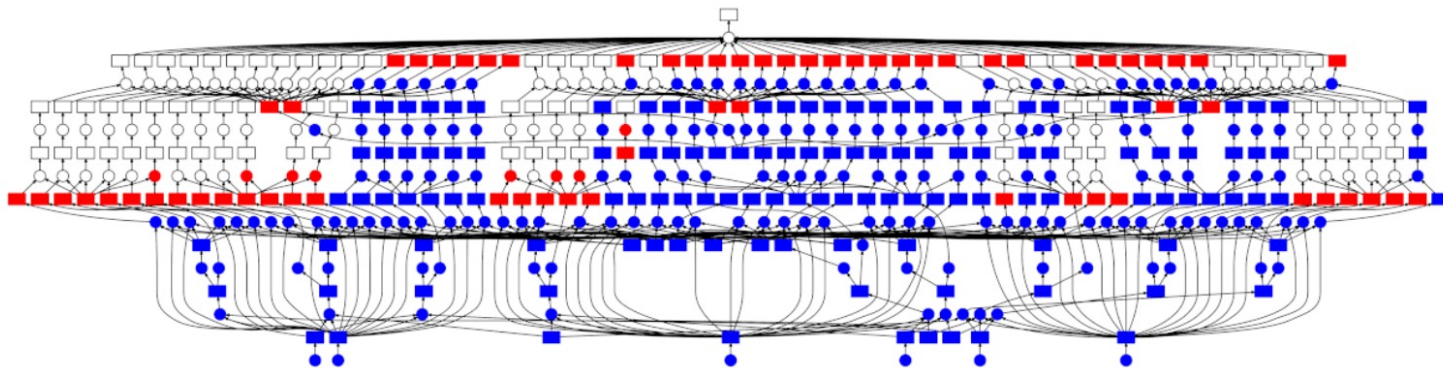
Créditos: <https://docs.rapids.ai/overview>

RAPIDS E DASK

- O RAPIDS e o Dask são integrados, portanto, o Dask nesse contexto é considerado um componente do RAPIDS.
- Muitos dos desenvolvedores RAPIDS da NVIDIA também são desenvolvedores Dask.
- Ao invés de utilizar um Dataframe Dask composto de Dataframes Pandas individuais, você pode ter um composto de Dataframes cuDF para executar em GPUs. Isso é possível porque eles seguem a mesma API no estilo do pandas.

DASK

- Dask permite que o trabalho seja distribuído criando uma representação Directed Acyclic Graph (DAG) de seu código em tempo de execução. Este gráfico é então passado para um scheduler que aloca tarefas individuais aos processos de trabalho. Esses processos de trabalho podem estar em uma única máquina, permitindo que você use todos os seus núcleos de CPU, ou em muitas máquinas, permitindo escalar para centenas ou até milhares de núcleos de CPU.





Créditos: <https://dask.org/>

DASK

- O Dask opera criando um cluster composto por um Cliente (Client) e vários “Workers”.
- O cliente é responsável pelo agendamento dos trabalhos;
- Os workers são responsáveis pela execução efetiva desse trabalho, possuindo basicamente duas funções:
 - Calcular tarefas conforme indicado pelo scheduler
 - Armazenar e fornecer resultados computados para outros workers ou clientes

```
INFO:distributed.worker:-----
INFO:distributed.scheduler:Register worker <WorkerState 'inproc://172.28.0.12/192/4', name: 0, s
INFO:distributed.scheduler:Starting worker compute stream, inproc://172.28.0.12/192/4
INFO:distributed.core:Starting established connection to inproc://172.28.0.12/192/5
INFO:distributed.worker:Starting Worker plugin shuffle
INFO:distributed.worker:      Registered to: inproc://172.28.0.12/192/1
INFO:distributed.worker:-----
INFO:distributed.core:Starting established connection to inproc://172.28.0.12/192/1
INFO:distributed.scheduler:Receive client connection: Client-1248026c-4a1c-11ee-80c0-0242ac1c00c
INFO:distributed.core:Starting established connection to inproc://172.28.0.12/192/6
```

**Client**
Client-1248026c-4a1c-11ee-80c0-0242ac1c000c
Connection method: Cluster object
Dashboard: <http://172.28.0.12:8787/status>
Cluster type: distributed.LocalCluster

**LocalCluster**
5e78011b
Dashboard: <http://172.28.0.12:8787/status>
Total threads: 2
Status: running
Workers: 1
Total memory: 12.68 GiB
Using processes: False

Cluster Info

Scheduler Info

DASK E RAPIDS

	Carrega dados maiores que a memória da CPU	Escala para múltiplas CPUs	Utiliza aceleração da GPU	Carrega dados maiores que a memória da GPU	Escala para múltiplas GPUs
Pandas	✗	✗	✗	✗	✗
Dask Dataframe	✓	✓	✗	✗	✗
RAPIDS (cuDF)	-	-	✓	✗	✗
RAPIDS (cuDF) + Dask Dataframe	-	-	✓	✓	✓

DASK

Há disponível diversos plugins ou integrações com o Dask, que o torna uma solução prática de deixar rodando em plataformas para gerenciamento e deploy de aplicações em cloud. A imagem abaixo mostra como é possível implementar com pouquíssimas linhas de código para a plataforma Kubernetes por exemplo.

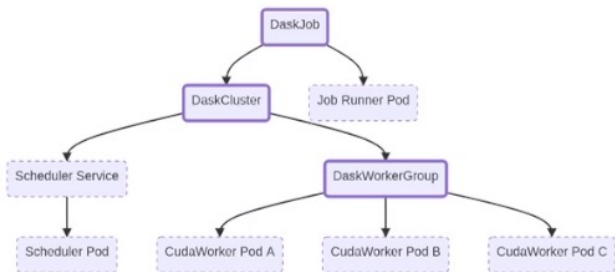
```
# Install dask-kubernetes
$ pip install dask-kubernetes

# Launch a cluster

>>> from dask_kubernetes.experimental \
    import KubeCluster

>>> cluster = KubeCluster(name="demo")

# List the DaskCluster custom resource that was created
for us under the hood
$ kubectl get daskclusters
NAME          AGE
demo-cluster  6m3s
```



```
[2]: from dask_kubernetes.experimental import KubeCluster

cluster = KubeCluster(name="rapids-dask",
    image="rapidsai/rapidsai-core:22.06-cuda11.4-runtime-ubuntu20.04-py3.9",
    worker_command="dask-cuda-worker",
    n_workers=2,
    resources={"limits": {"nvidia.com/gpu": "1"}},
    env={"DISABLE_JUPYTER": "true"})

cluster
```

Status **Scaling**

KubeCluster
rapids-dask

Dashboard: [notebook/kubeflow-user-example-com/rapids/proxy/rapids-dask-cluster-service.kubeflow-user-example-com/8767/status](#) **Workers:** 2

Total threads: 2 **Total memory:** 334.12 GiB

▼ Scheduler Info

Scheduler
Scheduler-7a70418f-c63c-4486-9243-6365160127da

Comm: [tcp://10.48.2.26:8786](#) **Workers:** 2

Dashboard: [notebook/kubeflow-user-example-com/rapids/proxy/10.48.2.26:8786/status](#) **Total threads:** 2

Started: 2 minutes ago **Total memory:** 334.12 GiB

▼ Workers

Worker: rapids-dask-cluster-default-worker-group-worker-2a9a6c8775
Comm: [tcp://10.48.1.25:41381](#) **Total threads:** 1

Dashboard: [notebook/kubeflow-user-example-com/rapids/proxy/10.48.1.25:36363/status](#) **Memory:** 167.08 GiB

Nanny: [tcp://10.48.1.25:35113](#)

Local directory: [/rapids/notebooks/dask-worker-space/worker-vh5ofxe3](#)

GPU: NVIDIA A100-SXM4-40GB **GPU memory:** 39.59 GiB

Worker: rapids-dask-cluster-default-worker-group-worker-5e564b6fb7

REFERÊNCIAS

Introdução

- <https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/>
- <https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/>
- <https://www.datanami.com/2019/03/22/how-walmart-uses-gpus-for-better-demand-forecasting>
- <https://www.nvidia.com/en-us/on-demand/session/gtcfall22-a41235/>
- <https://www.nvidia.com/en-us/on-demand/session/gtcfall22-a41386/>

RAPIDS

- <https://rapids.ai/ecosystem/#overview>
- cuDF - <https://docs.rapids.ai/api/cudf/stable/>
- cuML - <https://docs.rapids.ai/api/cuml/stable/>

Dask

- <https://docs.dask.org/en/stable/gpu.html>