HPS

Sören Metje

GPU Computing with Python

Table of contents

Motivation

Why Python?

- High-level programming
- Compatible with many platforms and systems
- Many high quality frameworks and libraries
- Widely distributed in many different domains
- Big community

Motivation

Why GPU Computing?

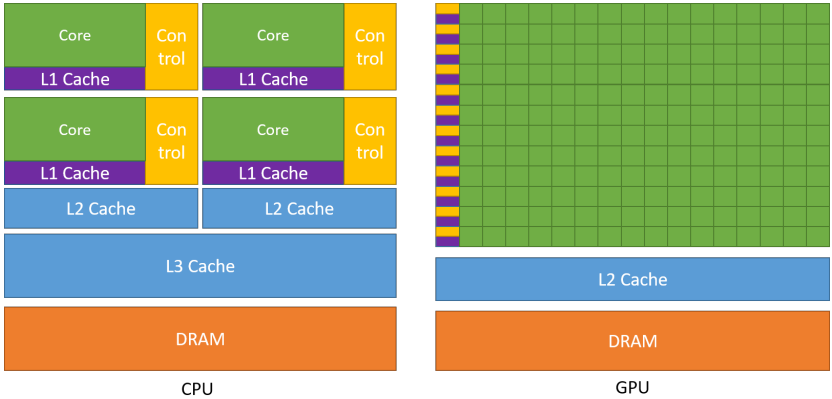
Reduce wall-clock time

Achieve higher cost-efficiency

Source: NVIDIA - GPU-Accelerated Google Cloud [NVId]

Outline

GPU Architecture



Source: CUDA Toolkit Documentation [NVIa]

Use Cases

When to use GPU Computing?

- Large data set available
- Parallel processing possible
- Use cases: Fluid dynamics, Image processing, Deep leaning, ...

When not to use GPU Computing?

- Data set is too small
- Data set is too big (exceeds GPU memory size)
- Large amount of small sequential operations

Source: NVIDIA - GPU-Accelerated Google Cloud [NVId]

CUDA

Definition

NVIDIA CUDA (Compute Unified Device Architecture)
is a parallel computing platform
and programming model for general computing on GPUs.

- Initial release: June 23, 2007
- Gives access to the GPU's virtual instruction set
- Enables execution of compute kernels
- Accessible through frameworks, libraries, and compiler directives
- Closed source

CUDA Compute Kernel

Definition

A compute kernel is a `function` compiled for accelerators (such as GPUs).

C++

```
ǟˢĪŇŕċòŇǟˢ IŕĴĒ ÞĖČ ĒĒʌĩŇŕòžʅ ɺ ĩŇŕòžʅ 7ɺ ĩŇŕòžʅ 8ʍ ʎ
    ĵŎž ĵ ˘ žıŲĖòĒ_Ēɯɺɯɼ
    8ʐĵʑ ˘ ʐĵʑ ˆ 7ʐĵʑɼ
Y

ĵŎž ōòĵŎʌʍ ʎ
    ǟ˞ ǟǟʷ
    ÞĖČ ĒĒǟǟψɘɺ vǟǟєʌ ɺ 7ɺ 8ʍɼ ǟ˞ ċŇŕČńŶ ůĖŲ ĪŲĴĒɺ žıŲĖòĒŶ ůĖŲ ċŇŕČń
Y
```

Compute Kernel Limitations

Allowed operations: basic math operations, if / else, for / while loops

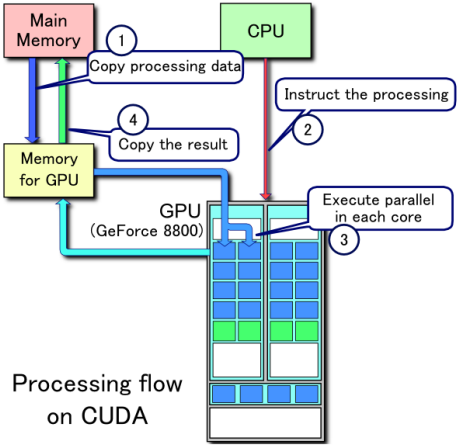
Can not explicitly return a value

Write results to passed array

C++

```
ə  ä$ÎŇŕċòŇä$ IŕĴĒ ÞĖČ ĒĒʌĩŇŕòžļ ɻ ĩŇŕòžļ 7ɻ ĩŇŕòžļ 8ʍ ʎ
ə      ĴŏžĴŏž ĵ ˇ žɪỤĖòĒ_ĒШɹШƒ
ə      8ƶĴƶ ˇ ƶĴƶ ^ 7ƶĴƶƒ
ε  ʏ
ɜ
ɜ  ĴŏžĴŏž õòĴŎʌʍ ʎ
ɞ      ä꞉ ä3ʷ
ɟ      ÞĖČ ĒĒäƷ̈ψɘɻ vä3̈ɛʌ ɻ 7ɻ 8ʍƒ  ä꞉ ċŇŕČńÝ ůĖỤ ĪỤĴĒɻ žɪỤĖòĒÝ ůĖỤ ċŇŕČń
ɠ  ʏ
```

CUDA Processing Flow



Source: Wikipedia - CUDA [Wika]

AMD ROCm

Definition

AMD ROCm (Radeon Open Compute) is a software stack for GPU programming.

"NVIDIA CUDA for AMD GPUs"

Initial release: November 14, 2016

Available on GitHub (open-source)

Supported by:

- PyTorch
- TensorFlow
- CuPy

Not as widely supported as NVIDIA CUDA

Gaining traction in the TOP500

TOP 500 List

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	1,110,144	151.90	214.35	2,942
4	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096

Source: TOP 500 List - June 2022 [TOP]

Outline

Python Frameworks for GPU Computing

Computing Frameworks

- Numba
- CuPy
- Scikit-cuda
- RAPIDS
- Triton (presented by Dimitris Oikonomou on 2022-05-19)
- ...

Deep Learning Frameworks

- PyTorch
- TensorFlow
- Keras
- ...

Python Frameworks for GPU Computing

Computing Frameworks

- Numba
- CuPy
- Scikit-cuda
- RAPIDS
- Triton (presented by Dimitris Oikonomou on 2022-05-19)
- ...

Deep Learning Frameworks

- PyTorch
- TensorFlow
- Keras
- ...

Numba

Definition

Numba is a just-in-time compiler
for numerical functions in Python.

Translates Python functions
to optimized machine code at runtime

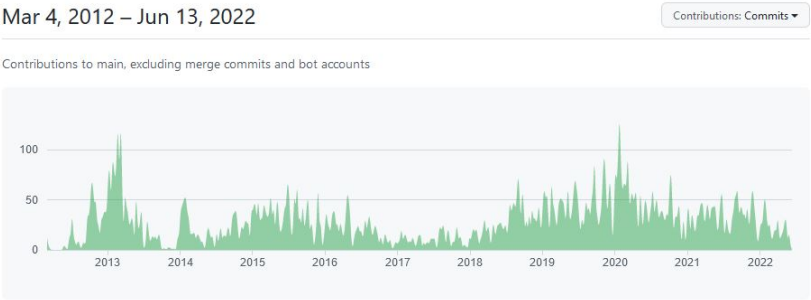
Compiles code for CPU and GPU

Supports

- NVIDIA GPUs: CUDA
- ~~AMD GPUs: ROCm~~ (deprecated)

Available on GitHub (open-source)

Numba - Development



Source: GitHub - Numba [Gita]

Numba - Programming Approaches

2 Approaches for GPU Programming

- Universal functions
- CUDA Kernels

Numba - Universal Functions

Definition

A universal function (or ufunc for short) is a `function` that operates on arrays in an element-by-element fashion.

Python

```
ĩŲŕō ŎƃōĊò Ĵōůŕųž lĖČžŕŲĴŖĖ

ƃIĖČžŕŲĴŖĖʌʐʠĩŇŕòžəəʌĩŇŕòžəəɻ ĩŇŕòžəəʍʠʑʍ
ĒĖĩ òĒĒɠƃĩƃŎČʌɯɻ ʃʍɼ
    ŲĖžƃŲŎ ɯ ˆ ʃ

Ŏ ˘ ɘdddd
ò ˘ ŎůɹòŲòŎĪĖʌŎʍɹòŶžʃůĖʌŎůɹĩŇŕòžəəʍ
Ċ ˘ ɘ ɿ ò
ŕƃž ˘ òĒĒɠƃĩƃŎČʌòɻ Ċʍ
```

Numba - Universal Functions on GPU

Python

```
ĩŲŕō ŏƃōċò Ĵōůŕųž ǀĖČžŕŲĴŖĖ

ƃIĖČžŕŲĴŖĖʌʐʧĩŇŕòžəəʌĩŇŕòžəəɻ ĩŇŕòžəəʍʧʑɻ žòŲĪĖž ʧČƃĒòʧʍ
ĒĖĩ òĒĒɕƃĩƃŎČʌɯɻ Ɲʍɼ
    ŲĖžƃŲŎ ɯ ˆ Ɲ
```

1. Compile CUDA kernel
2. Allocate GPU memory
3. Copy data to the GPU
4. Executed CUDA kernel
5. Copy result back to the CPU
6. Return the result

Numba - CUDA Kernels

Python

```
ĪŲŕō ŎƃōċÒ ĴōůŕŲž ČƃĒò
ĴōůŕŲž ŎƃōůƝ òŶ Ŏů

БČƃĒòɺŁĴž
ĒĖĪ òĒĒɢńĖŲŎĖŇʌɯɻ Ɲɻ ŕƃžʍɼ
    ŶžòŲžɻ ŶžŲĴĒĖ ˘ ČƃĒòɺĪŲĴĒʌɘʍɻ ČƃĒòɺĪŲĴĒŶĴŖĖʌɘʍ ʃ ɘ ˘ ŕŎĖ ĒĴōɺ žɪŲĖòĒ ĪŲĴĒ
    ĪŕŲ Ĵ ĴŎ ŲòŎĪĖʌŶžòŲžɻ ɯɺŶɪòůĖʑɗʑɻ ŶžŲĴĒĖʍɼ
        ŕƃžʑĴʑ ˘ ɯʑĴʑ ˆ Ɲʑĵʑ

ɯ ˘ ŎůɺòŲòŎĪĖʌɘɗɗɗɗɗʍɺòŶžƝůĖʌŎůɺĪŇŕòžɘɘʍ
Ɲ ˘ ŎůɺòŲòŎĪĖʌɘɗɗɗɗɗʍɺòŶžƝůĖʌŎůɺĪŇŕòžɘɘʍ
ŕƃž ˘ ŎůɺĖōůžƝɢŇĴńĖʌɯʍ
òĒĒɢńĖŲŎĖŇʑɘɗɻ ɘɘɻʑʌɯɻ Ɲɻ ŕƃžʍ ʃ ċŇŕČńŶ ůĖŲ ĪŲĴĒɻ žɪŲĖòĒŶ ůĖŲ ċŇŕČń
ůŲĴŎžʌŕƃžʍ ʃ ʑɗɺɗ ɘɺɗ ɛɺɗ ɜɜʷ ɘɺɠɠɠɻĖ̂ ɗɜʑ
```

CuPy

Definition

CuPy is a NumPy/SciPy-compatible array library for GPU-accelerated computing.

"NumPy for GPU computing"

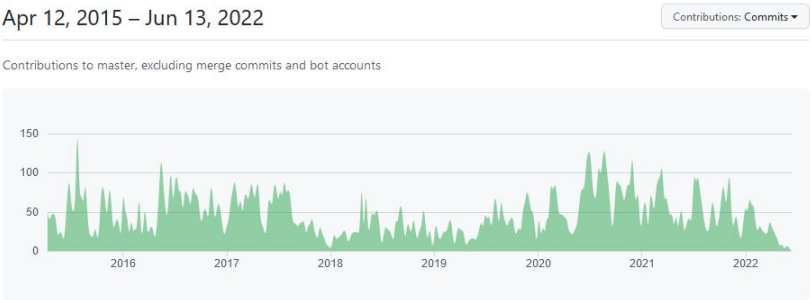
Provides various math operations

Supports

- NVIDIA GPUs: CUDA
- AMD GPUs: ROCm (experimental)

Available on GitHub (open-source)

CuPy - Development



Source: GitHub - CuPy [Gitb]

CuPy - Math Functions

Python

```
Ĵōůŕųž ČƃůƝ òŶ Čů

ɯ ˘ Čů.òųòŎĪĖʌəďďďďďʍ
Ɲ ˘ ɯ ʅ ə

ŕƃžə ˘ Čů.òĒĒʌɯɻ Ɲʍ ʆ òųųòƝʌʐ    ďɻ    ə˞ɻ    ɝɻ ɜ̈ɜ̈ʷɻ əɠɠɠɞʑʍ
ŕƃžə ˘ Čů.ŶƃōʌɯʍʆòųųòƝʌɞďɛɠɟəɞďɛʍ
ŕƃžə˞ ˘ Čů.ŇĴŎòŇĪ.ŎŕųōʌɯʍʆòųųòƝʌəɟəɜɞəɟəɺɝɜəɟďɠəəʍ
ʆ ɜ̈ɜ̈ʷ
ʆ  ŇŇ Čŕōōŕ Ŏ vƃō·Ɲ ĩƃŎČžĴŕŎŶ òųĖ Ŷƃůŕųžė Ē ċƝ 8ƃ·Ɲ
```

CuPy - CUDA Kernels

Python

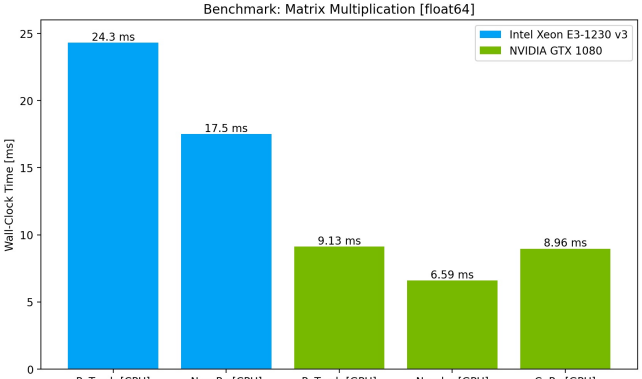
```
ĵōůŕŲž ČƃůƝ òŶ Čů

òĒĒ ˘ Čů.ɺºòɪmĖŲŎĖŇʌŲţţ
ĖɯžĖŲŎ ʦ8ʦ ӟˢĪŇŕċòŇӟˢ lŕĵĒ òĒĒʌČŕŎŶž ĵŎžɿ ůɻ ČŕŎŶž ĵŎžɿ űɻ ĵŎžɿ Ʀᴍ ʎ
    ĵŎž žĵĒ ˘ ċŇŕČń>ĵōɺɯ ɿ ċŇŕČń_ĒɯɺɯˆžɪŲĖòĒ_ĒɯɺɯΓ
    ƦȥžĵĒȥ˘ ůȥžĵĒȥˆ űȥžĵĒȥΓ
 Y
ţţţɻ ţòĒĒţᴍ
ɯ ˘ Čů.ɺòŲòŎĪĖʌɘɗɗɗɗɗɻ ĒžƝůĔ ĵŎžᴍ
Ɲ ˘ Čů.ɺòŲòŎĪĖʌɘɗɗɗɗɗɻ ĒžƝůĔ ĵŎžᴍ
ŕƃž ˘ Čů.ɺƦĖŲŕŶʌɘɗɗɗɗɗɻ ĒžƝůĔ ĵŎžᴍ
òĒĒʌʌɘɜɗɻ ɘᴍɻ ʌɘɗɘɛɻ ɘᴍɻ ʌɯɻ Ɲɻ ŕƃžᴍᴍ ʆ ċŇŕČńŶ ůĖŲ ĪŲĵĒɻ žɪŲĖòĒŶ ůĖŲ ċŇŕČń
ůŲĵŎžʌŕƃžᴍ ʆ òŲŲòƝʌȥ ɗ ɘ ɛ ӟӟʷ ɘɠɠɠɛ ɘɠɠɠɜ ɘɠɠɠɟȥᴍ
```

Benchmark - Matrix Multiplication

Code for benchmarking is available on GWDG GitLab

Benchmark - Matrix Multiplication Float64



Wall-clock time in milliseconds for computing result of fast matrix multiplication. Mean of 100 loops and 4 runs.
Multiplied a 1024x2048 matrix and a 2048x512 matrix of float64.
Measured time does not count in time to copy matrices to GPU and result matrix back from GPU.

Code for benchmarking is available on GWDG GitLab

Benchmark - Matrix Multiplication

Code for benchmarking is available on GWDG GitLab

Benchmark - Matrix Multiplication

Code for benchmarking is available on GWDG GitLab

Outline

New Trends

What's new?

- GPUs get more powerful
- More complex models and computations
- New high-level Python frameworks provide features for various use cases
- ROCm (open-source) is gaining traction

Summary

GPUs

- achieve massive data parallelism
- reduce wall-clock time
- increase cost efficiency

Summary

GPUs

- achieve massive data parallelism
- reduce wall-clock time
- increase cost efficiency

Numba Advantages

- Enables implementing own universal functions & kernels in Python running on GPU

Summary

GPUs

- achieve massive data parallelism
- reduce wall-clock time
- increase cost efficiency

Numba Advantages

- Enables implementing own universal functions & kernels in Python running on GPU

CuPy Advantages

- Easily move existing NumPy code towards GPU computing
- Directly use NumPy-style array operations and execute them on GPU
- Great starting-point to learn about GPU computing

References I

Avimanyu Bandyopadhyay. *Hands-On GPU Computing with Python*. Packt Publishing Ltd, 2019. ISBN: 9781789341072.

NVIDIA. *Programming Guide :: CUDA Toolkit Documentation*. URL: ıžžůŶɼʈʈĒŕČŶɹŎlĴĒĴòɹČŕōʈČƃĒòʈČƃĒòʖČʖůŲŕĪŲòōōĴŎĪʖĪƃĴĒĖʈĴŎĒĖɯɹıžōŇ.

NVIDIA. *PTX ISA :: CUDA Toolkit Documentation*. URL: ıžžůŶɼʈʈĒŕČŶɹŎlĴĒĴòɹČŕōʈČƃĒòʈůòŲòŇŇĖŇʖžıŲĖòĒʖĖɯĖČƃžĴŕŎʈĴŎĒĖɯɹıžōŇ.

NVIDIA. *CUDA Zone - Library of Resources | NVIDIA Developer*. URL: ıžžůŶɼʈʈĒĖlĔŇŕůĖŲɹŎlĴĒĴòɹČŕōʈČƃĒòʖʀŕŎĖ.

NVIDIA. *GPU-Accelerated Google Cloud*. URL: ıžžůŶɼʈʈɪɪɪɹŎlĴĒĴòɹČŕōʈĖŎʖƃŶʈĒòžòʖČĖŎžĖŲʈĪůƃʖČŇŕƃĒʖČŕōůƃžĴŎĪʈĪŕŕĪŇĖʖČŇŕƃĒʖůŇòžĪŕŲōʈ.

Wikipedia. *CUDA*. URL: ıžžůŶɼʈʈĖŎɹɪĴńĴůĖĒĴòɹŕŲĪʈɪʈĴŎĒĖɯɹůıůʁžĴžŇĔ 8Ĕ> BŕŇĒĴĔ ɘɗɻɠɻɠɠɘɘɻ.

Wikipedia. *ROCm*. URL: ıžžůŶɼʈʈĖŎɹɪĴńĴůĖĒĴòɹŕŲĪʈɪʈĴŎĒĖɯɹůıůʁžĴžŇĔ º|8ōBŕŇĒĴĔ ɘɗɻɜɛɘɻɗɘɛ.

Wikipedia. *Compute kernel*. URL: ıžžůŶɼʈʈĖŎɹɪĴńĴůĖĒĴòɹŕŲĪʈɪʈĴŎĒĖɯɹůıůʁžĴžŇĔ 8ŕōůƃžĖɠńĖŲŎĖŇBŕŇĒĴĔ ɘɗɗɻɘɜɞɛɠɘ.

Microsoft. *VISC: Virtual Instruction Set Computing*. URL: ıžžůŶɼʈʈɪɪɪɹɲŕƃžƀĖɹČŕōʈɪòžČıʁĽ ɯuɠlB|\ĩɜŎ_.

References II

GitHub. *Numba*. URL: ıžžůŶɼţţĪĴžıƃċɺČŕōţŎƃōċòţŎƃōċò.

Anaconda. *Numba: A High Performance Python Compiler*. URL: ıžžůŶɼţţŎƃōċòɺůŊĒòžòɺŕŲĪţ.

Anaconda. *Numba documentation*. URL: ıžžůŶɼţţŎƃōċòɺŲĖòĒžıĖĒŕČŶɺĴŕţĖŎţŶžòċŇĖţĴŎĒĖɯɺıžōŇ.

Villoro. *Villoro - numba*. URL: ıžžůŶɼţţŀĴŇŇŕŲŕɺČŕōţůŕŶžţŎƃōċò.

GitHub. *CuPy*. URL: ıžžůŶɼţţĪĴžıƃċɺČŕōţČƃůŊţČƃůŊ.

Preferred Networks, Inc. and Preferred Infrastructure, Inc. *CuPy NumPy & SciPy for GPU*. URL: ıžžůŶɼţţĒŕČŶɺČƃůŊɺĒĖŀţĖŎţŶžòċŇĖţĴŎĒĖɯɺıžōŇ.

NumPy. *NumPy documentation*. URL: ıžžůŶɼţţŎƃōůŊɺŕŲĪţĒŕČţŶžòċŇĖţĴŎĒĖɯɺıžōŇ.

Mindfire Solutions. *Python: 7 Important Reasons Why You Should Use Python*. URL: ıžžůŶɼţţōĖĒĴƃōɺČŕōţƃōĴŎĒĪĴŲĖŶŕŇƃžĴŕŎŶɺƃŶòţůŊžıŕŎɹɞɹĴōůŕŲžòŎžɹŲĖòŶŕŎŶɹɪıŊɹŊŕƃɹŶıŕƃŇĒɹƃŶĖɹůŊžıŕŎɹɜɺɗɘòɠɺòɗĒɗċ.

Marko Aleksic. *CPU Vs. GPU: A Comprehensive Overview*. URL: ıžžůŶɼţţůıŕĖŎĴɯŎòůɺČŕōţńċţČůƃɹŀŶɹĪůƃ.

References III

TOP500. *TOP 500 List*. URL: ıžžůŶɼţţĪĪĪɹžŕů3ďďɹŕŲĪţŇĴŶžŶţžŕů3ďďţŇĴŶžţəďəəţďɘţ.

Amazon Web Services. *Amazon EC2 P3 instances*. URL: ıžžůŶɼţţòĪŶɹòōòʀŕŎɹČŕōţĖČəţĴŎŶžòŎČĖɔžƝůĖŶţůəţʁŎČɚ ıɞŇŶ.

Google. *Google Colab*. URL: ıžžůŶɼţţČŕŇòċɹŲĖŶĖòŲČıɹĪŕŕĪŇĖɹČŕōţ.

GPU as a Service

Instance Size	GPUs - Tesla V100	GPU Peer to Peer	GPU Memory (GB)	vCPUs	Memory (GB)	Network Bandwidth	EBS Bandwidth	On-Demand Price/hr*	1-yr Reserved Instance Effective Hourly*	3-yr Reserved Instance Effective Hourly*
p3.2xlarge	1	N/A	16	8	61	Up to 10 Gbps	1.5 Gbps	$3.06	$1.99	$1.05
p3.8xlarge	4	NVLink	64	32	244	10 Gbps	7 Gbps	$12.24	$7.96	$4.19
p3.16xlarge	8	NVLink	128	64	488	25 Gbps	14 Gbps	$24.48	$15.91	$8.39
p3dn.24xlarge	8	NVLink	256	96	768	100 Gbps	19 Gbps	$31.218	$18.30	$9.64

Example Pricing: Amazon EC2 P3 instances. Source: AWS [Ama]

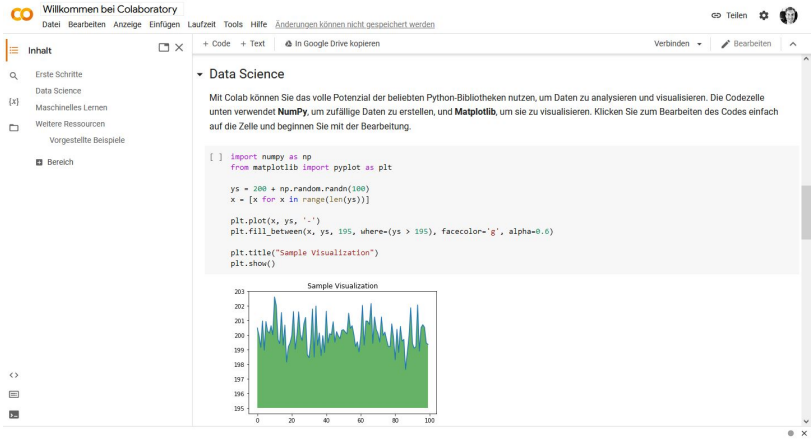
Advantages

- Scalable resources
- On-demand pricing

Disadvantages

- Expense for large periods of time
- Data confidentiality not guaranteed depending on vendor

GPU as a Service



Google Colab Jupyter Notebooks. Source: Google [Goo]

NVIDIA PTX

Definition

NVIDIA PTX (Parallel Thread Execution) is a low-level parallel thread execution virtual machine and instruction set architecture (ISA)

- Exposes the GPU as a data-parallel computing device
- Interprets compiled code (analog to Java byte code interpreted by JVM)

Advantage

- Achieves portability of source code among multiple NVIDIA GPUs