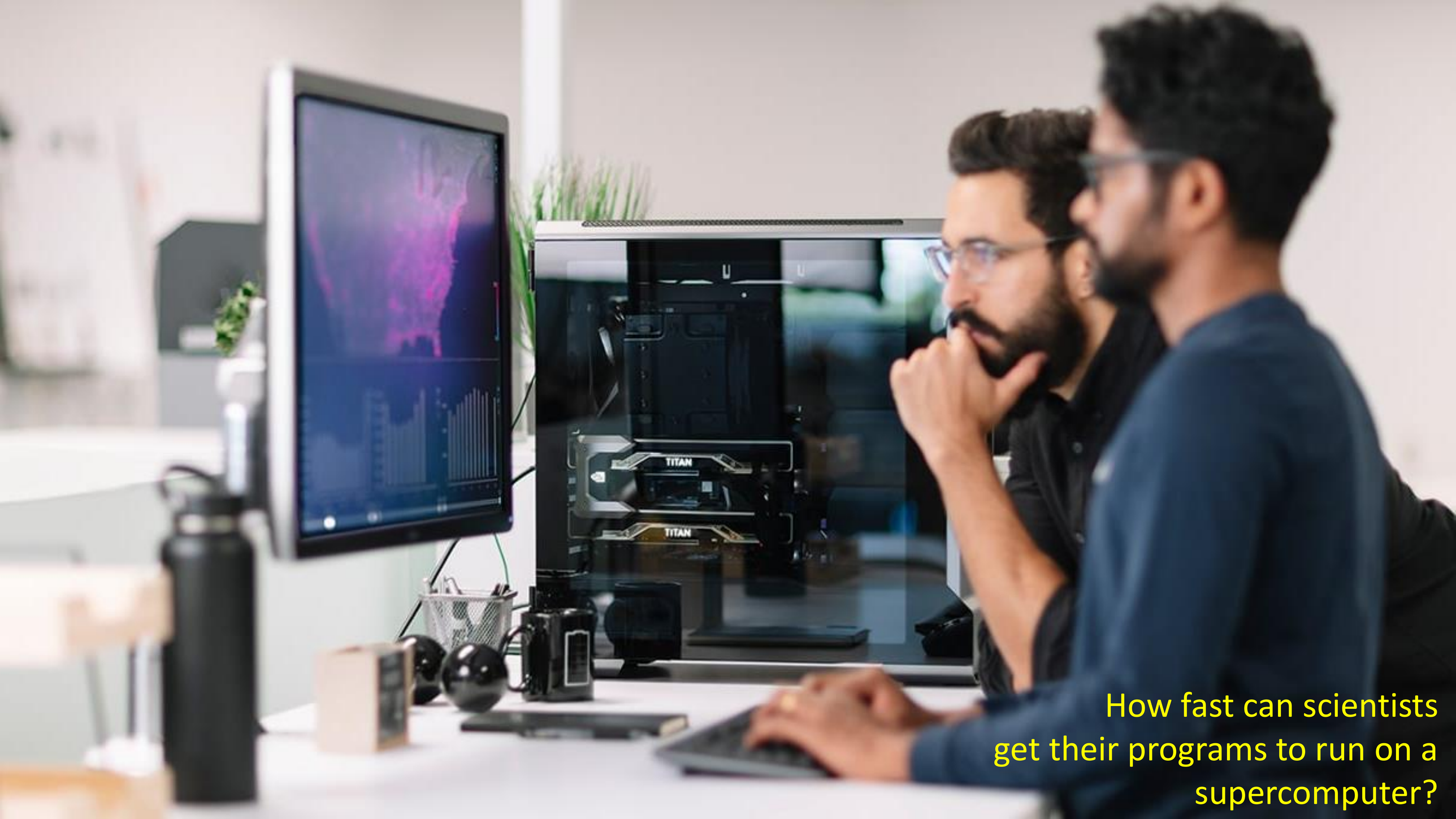
Seamlessly scale your python program from single CPU core to multi-GPU multi-node HPC cluster with cuPyNumeric

Bo Dong, Principal Technical Product Manager

Manolis Papadakis, Software Engineering Manager



How fast can scientists get their programs to run on a supercomputer?

Currently using NumPy extensively + some Dask. "Big gap" between what they're doing now and where they want to be. Ultimate goal is to look at the goldmine of data, many TB's and analyze with their methods.

Our objective is not to reach 100% performance in ALL cases (which is anyways not possible), but to provide fast scripting + reasonable performance for complex SciML workflows.

A scientist can only get 2-3 days of beam time each year for experiments. They must run the experiments, analyze the data, adjust and run experiments again. They are in the mode of using libs like NumPy to write their analysis and very unfamiliar with writing code to handle the complexity of GPU or distributed systems.

The use case is to process xRAY CT data. With NumPy on the CPU he cannot go beyond 500 cubes. However, the problem space could require the whole Venado to run. The top challenge is how to scale from CPU to GPU to multi-GPU to multi-node.

“

With Venado online, researchers will have much more GPU resources available. We need to enable them to fully utilize the resources.

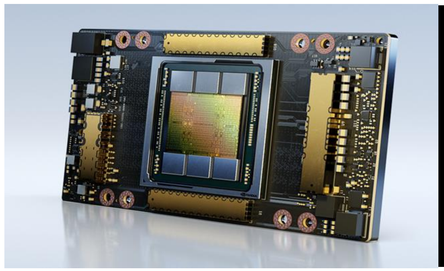
”

What is cuPyNumeric?

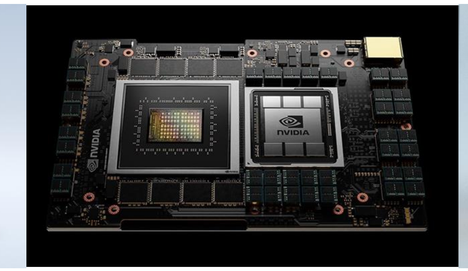
- Standard Python programming w/o constraints.
- Transparent scaling from single CPU core to a multi-GPU multi-Node supercomputer.

Supercomputer

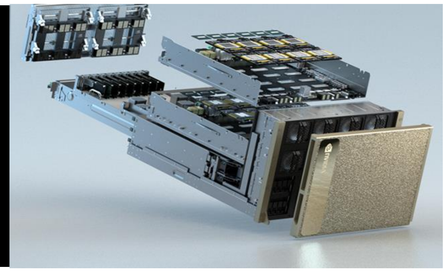
DGX Cloud



GPU



Grace Hopper Superchip



DGX

interchangeable

```
import cupynumeric as np
#import numpy as np

A = np.random.rand(M, K)
B = np.random.rand(K, N)
C = A @ B

print(C)
print(type(C))
```

No code change to run on Multi-GPU Multi-Node

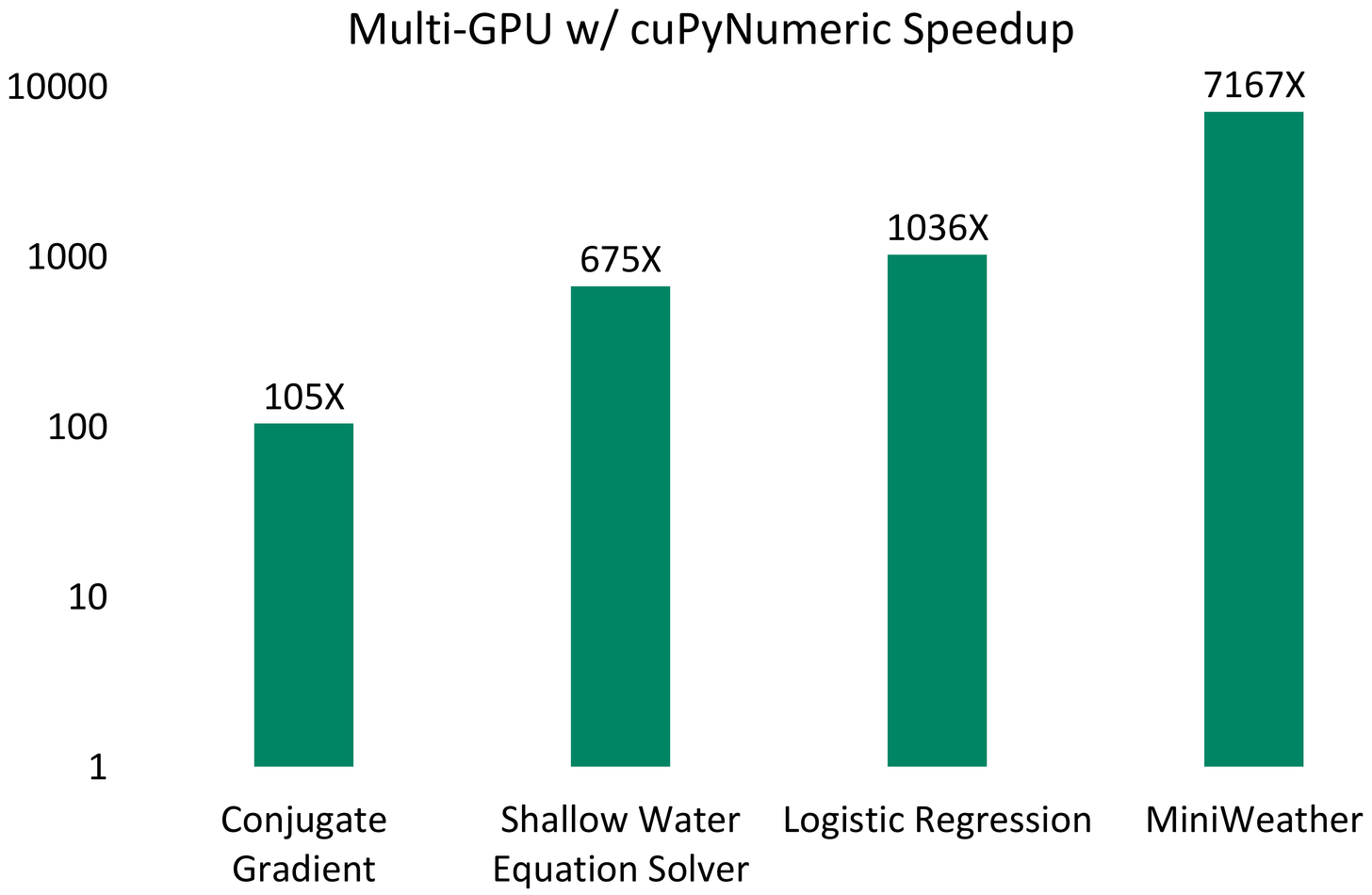
```
(legate) bod@dgx-1:~$ legate --gpus 1 foo.py

(legate) bod@dgx-1:~$ legate --gpus 8 foo.py

(legate) bod@dgx-1:~$ legate --nodes 64 --gpus 8 foo.py
```

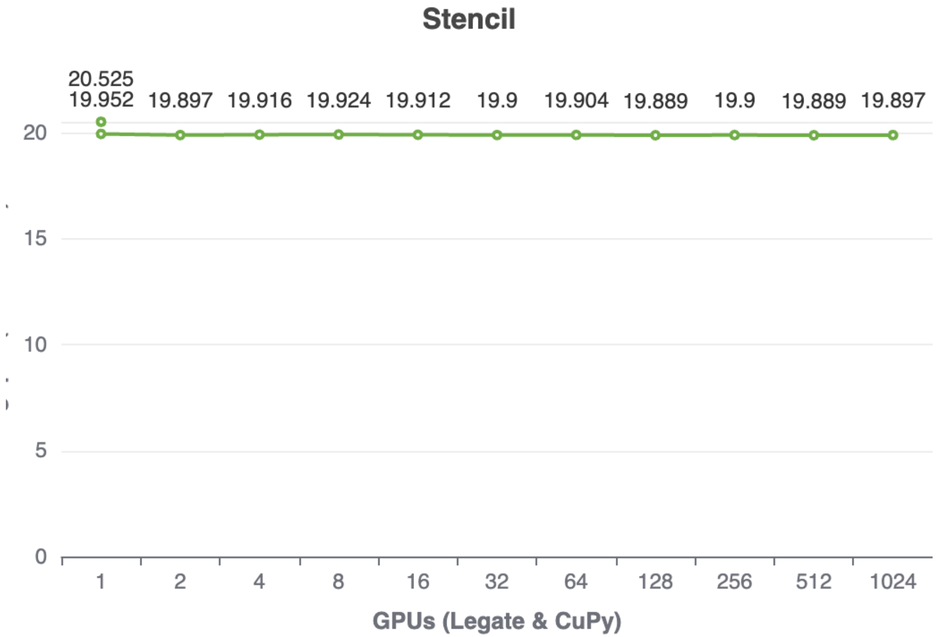
Live Demo

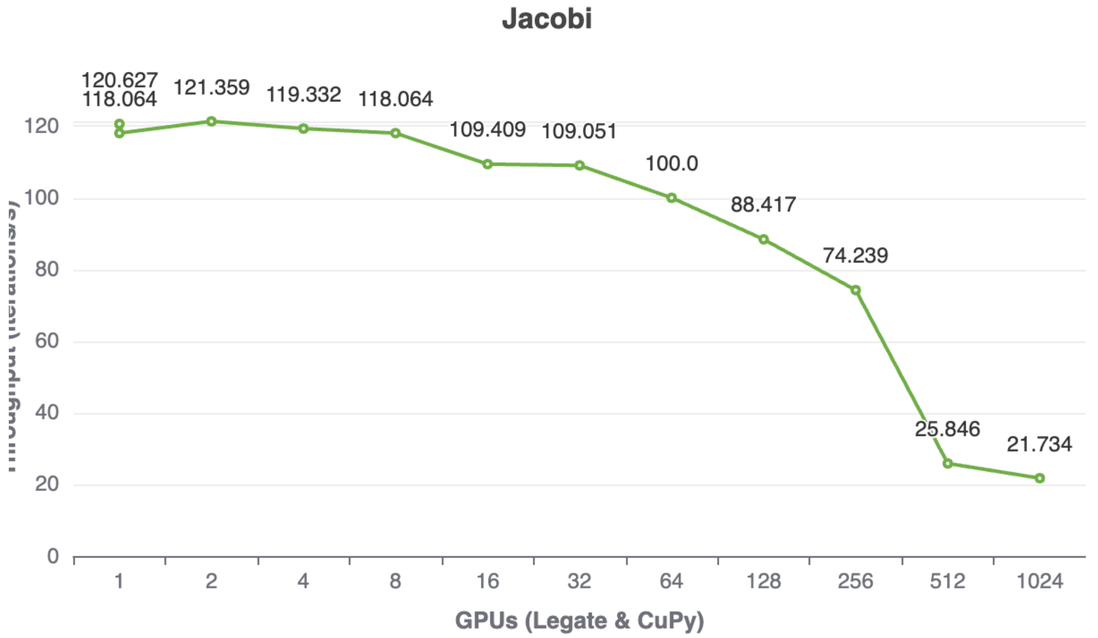
A Not So Fair Comparison

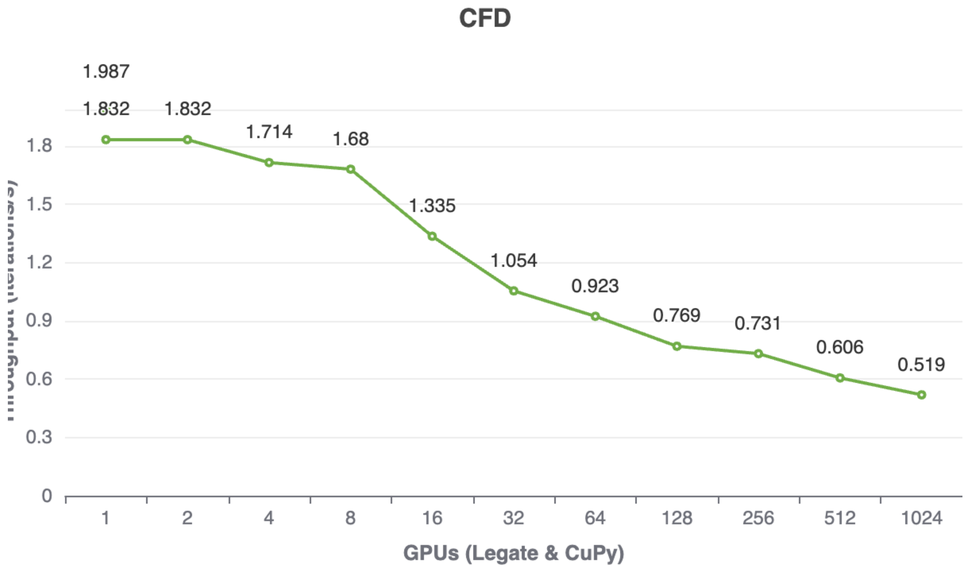


- It’s unfair because the number is single CPU vs 8 GPUs.
- It’s also workload dependent.
- In the real-world use cases the speedup will be less end to end.
- It’s still relevant because:
  - The runs are same code w/o any change.
  - NumPy can only use one CPU whereas cuPyNumeric scales.

Scale to 1000+ GPUs



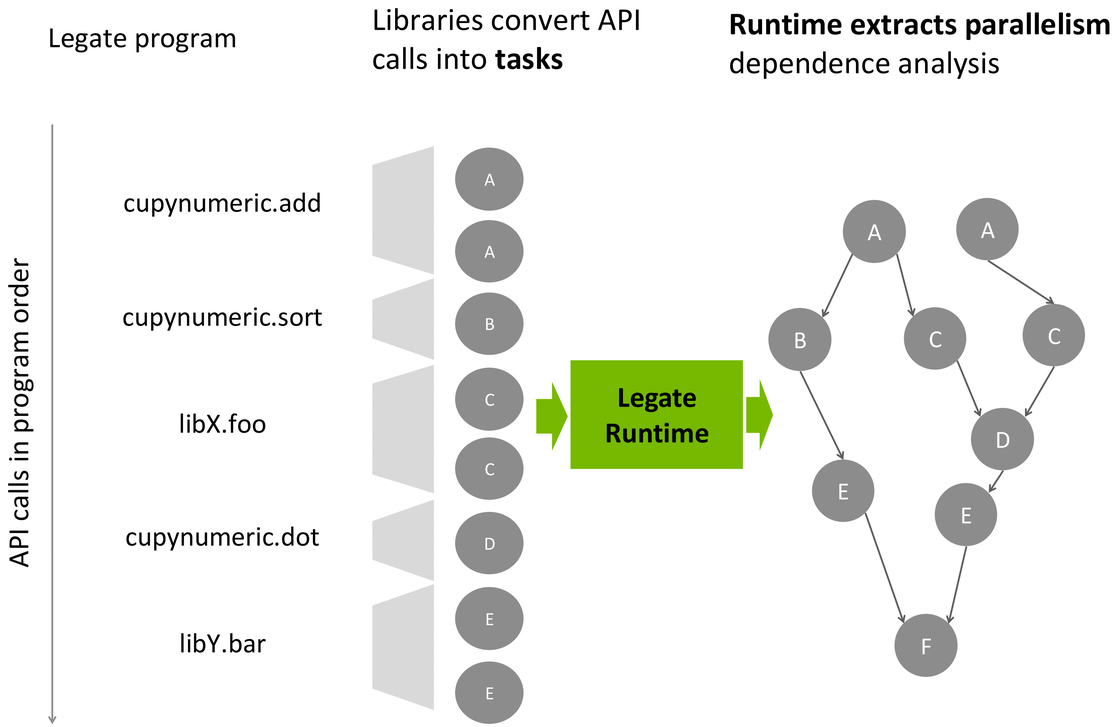




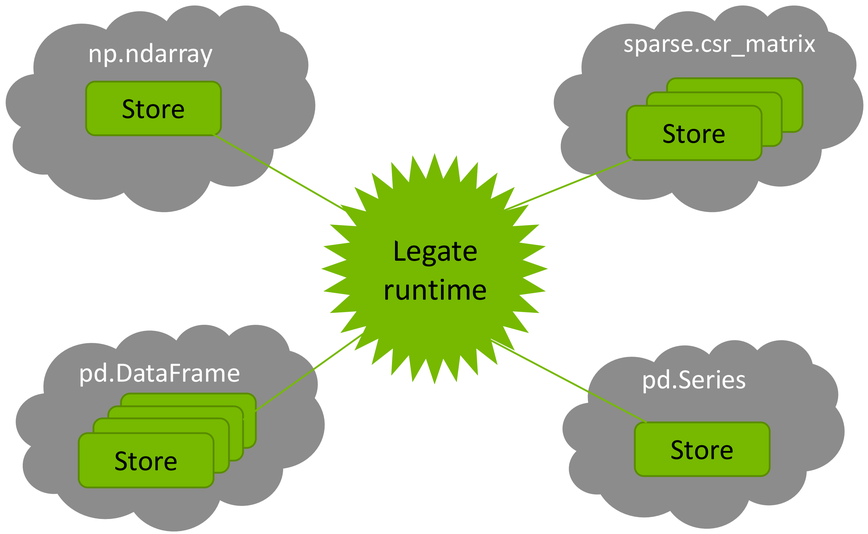
Growing the input size each time such that each GPU has the same working set size.
On EOS w/ 8 H100 per node.

Legate -- cuPyNumeric's Secret Sauce

Extract Implicit Parallelism for Effortless Scaling



Unified Data Abstraction for Composability



Call To Action

https://nvda.ws/3AVtGtx

Please give it a try!

Your feedback to legate@nvidia.com is highly appreciated.

Your voice matters!