Legate & cuNumeric

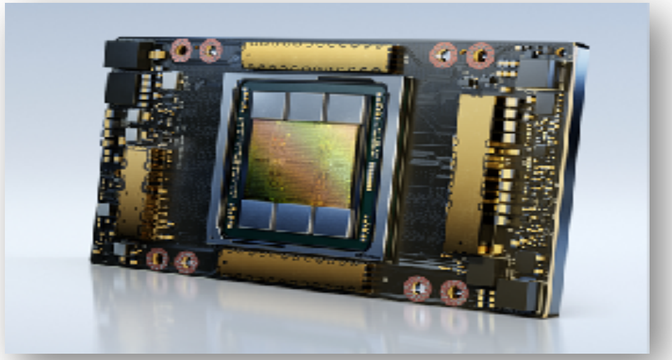
Rohan Yadav | PAW-ATM'23 | November 13, 2023

Mission: Accessible Accelerated Computing

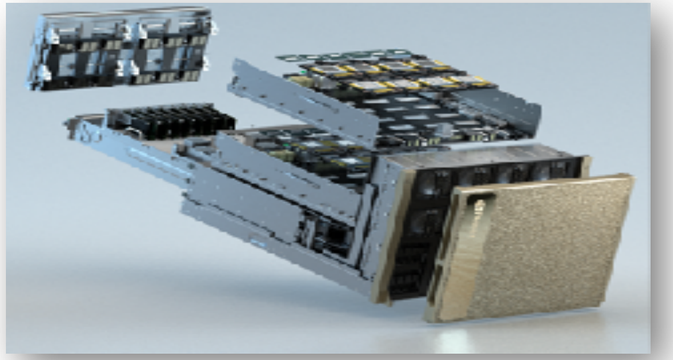
Accelerated computing with no pain of complex distributed programming

```
# Generate a random positive semi-definite matrix
A = scipy.sparse.random(n, n, format="csr")
A = 0.5 * (A + A.T) + n * scipy.sparse.eye(n)
# Estimate the maximum eigenvalue of A
x = numpy.random.rand(A.shape[0])
for _ in range(iters):
  x = A @ x
  x /= numpy.linalg.norm(x)
result = numpy.dot(x.T, A @ x)
```

Grace Hopper Superchip



GPU

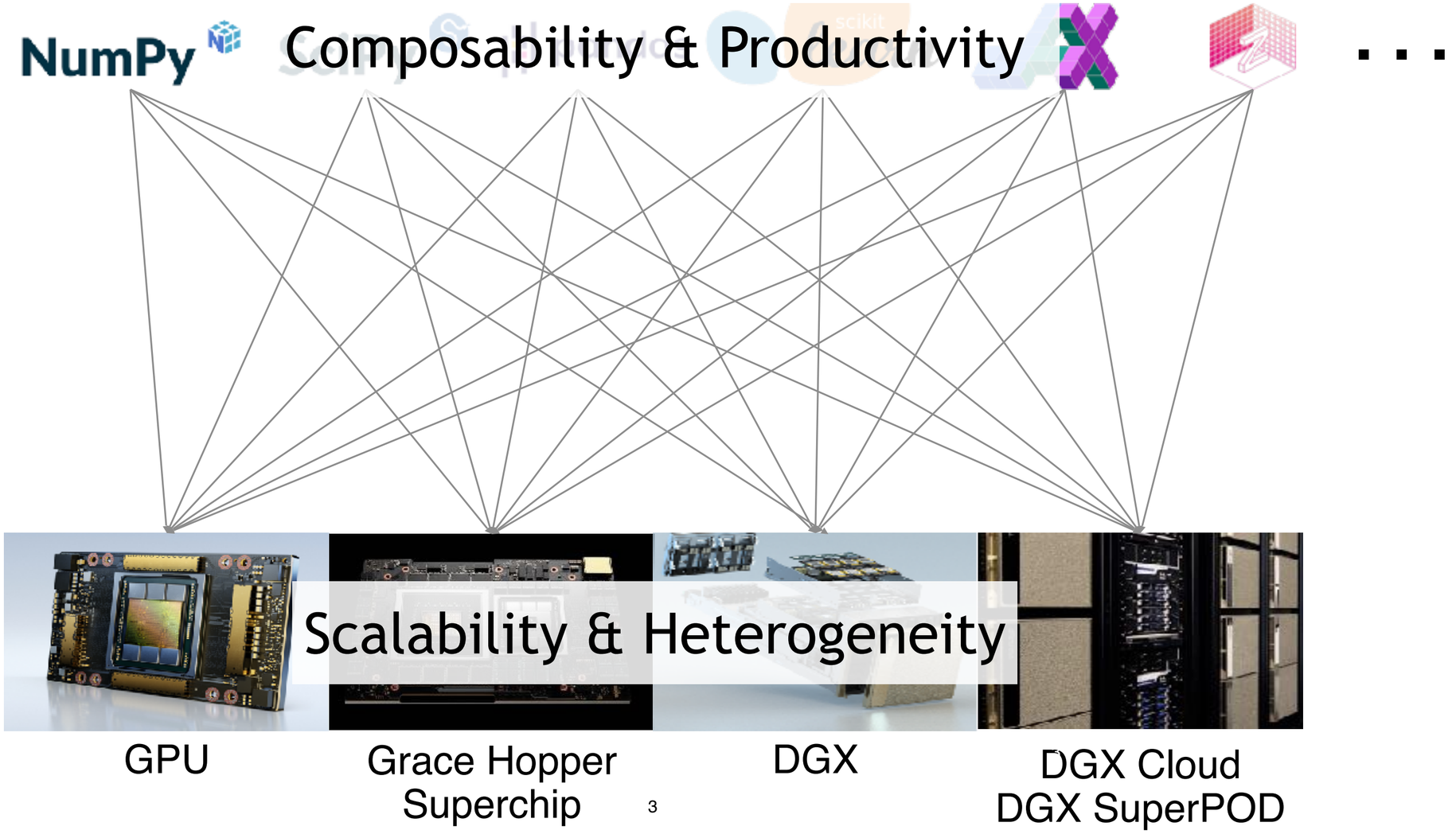


DGX

DGX Cloud
DGX SuperPOD

Challenges for Library Developers

Problem domains with accelerated computing needs



Enter Legate

Common scalable runtime software stack for NVIDIA hardware

Composable Legate Libraries

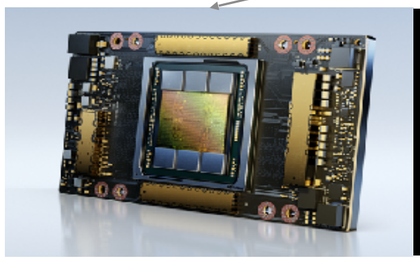
cuNumeric | Legate Sparse | Legate Pandas | Legate ML | Legate JAX | Legate IO | . . .

Legate Runtime Stack

Legate - Productivity & Composability Layer

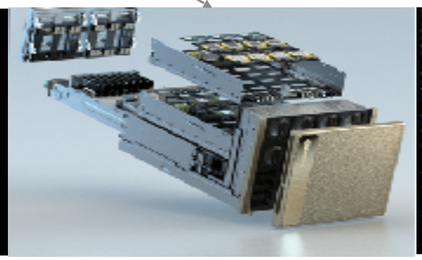
Legion - Implicit Parallelism Layer

Realm - Runtime for Scalable and Portable Execution



GPU

Grace Hopper Superchip



DGX

DGX Cloud
DGX SuperPOD

Legate's Secret Sauce

Extract Implicit Parallelism for Effortless Scaling

Legate program

API calls in program order

Libraries only convert API calls into **tasks**

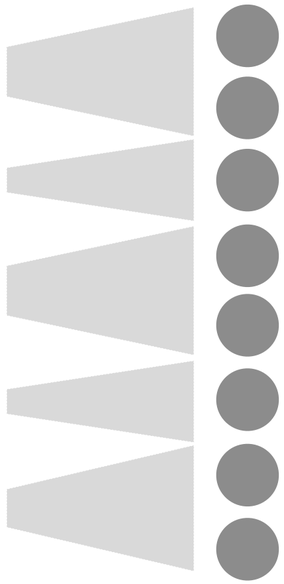
cunumeric.add

cunumeric.sort

libX.foo

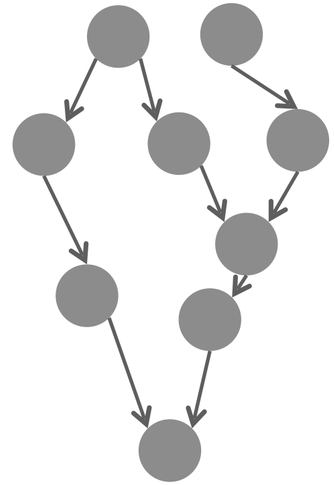
cunumeric.dot

libY.bar



Legate Runtime

Runtime extracts parallelism via dependence analysis



- Libraries are **free of explicit synchronization and data movement**, making them **composable** with each other
- Runtime weaves tasks from **multiple libraries** into a single execution with **all available parallelism** extracted
- Runtime inserts **minimum required data movement and synchronization**

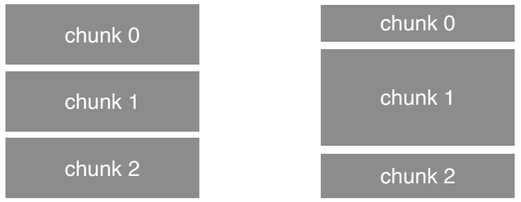
Legate's Secret Sauce

Unified Data Abstraction for Composability

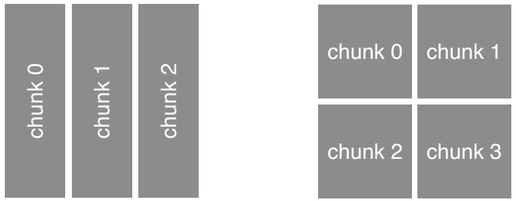
Composability challenge: cunumeric.nonzero

- Must return indices of non-zero entries in C order
- Wants inputs partitioned on only the first dimension

These inputs can be reused



These inputs **need repartitioning**

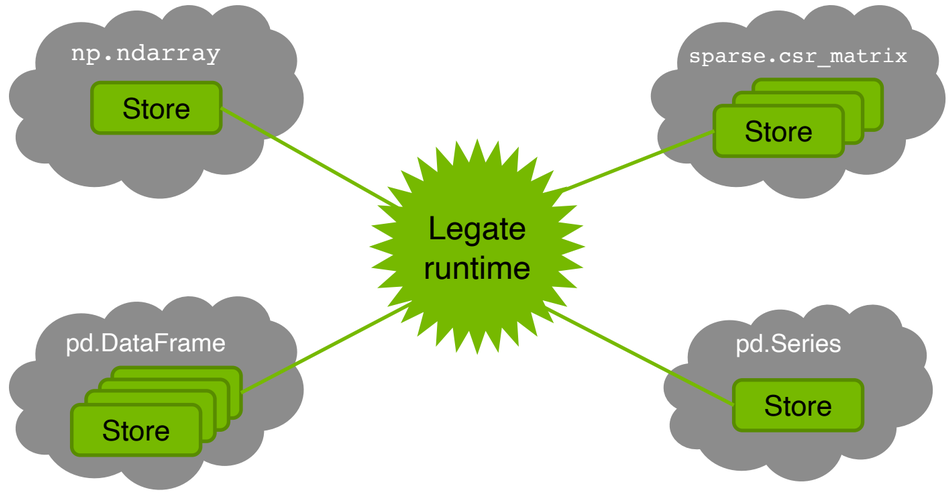


Input possibly produced by another library

- What kinds of partitions does it use?
- How do I convert between them?
- Can I do it without blocking?

Legate's solution:

- Libraries implement domain-specific containers on top of **Legate stores**
  - Global views to the data
  - Partition information maintained by the runtime

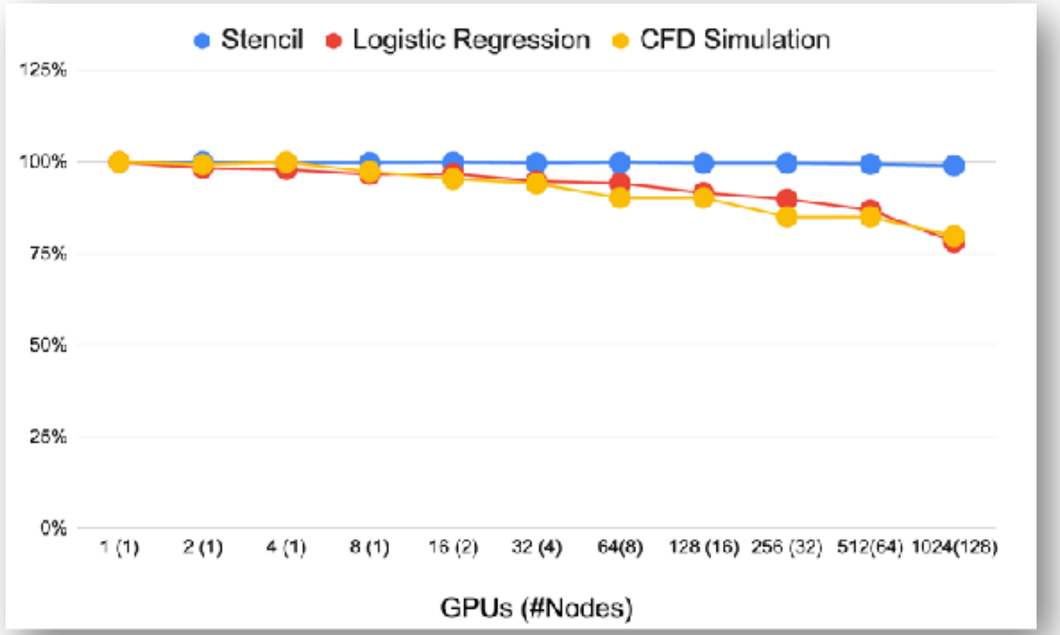


- Libraries specify **partitioning constraints** on tasks
  - Runtime infers if existing partition can be reused
  - Runtime builds data exchange graph, executes asynchronously

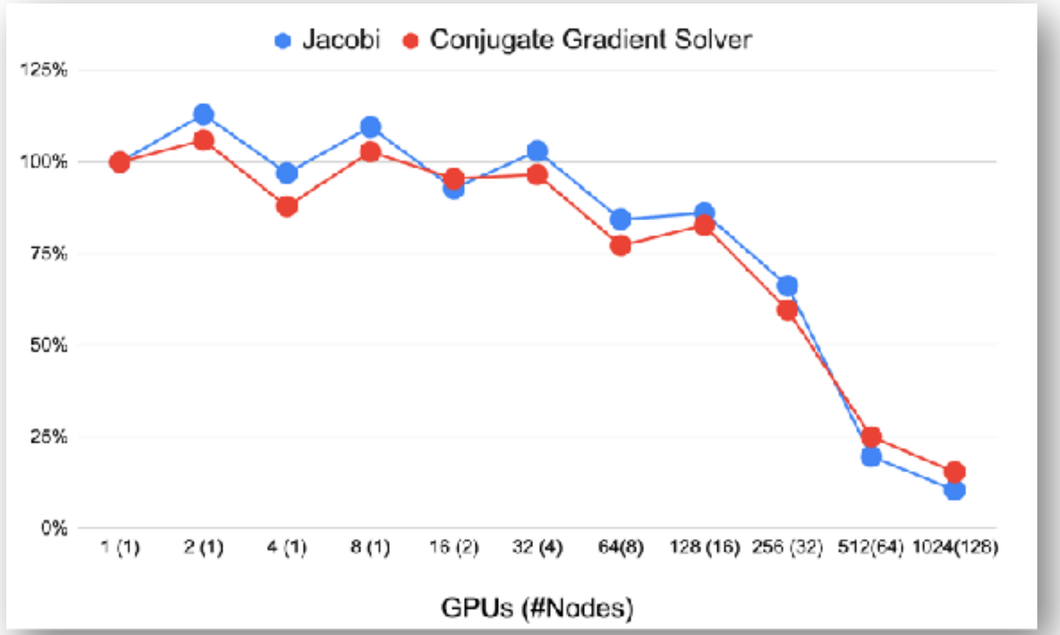
Out-of-the-Box Scaling for End-Users

cuNumeric: Good weak scaling across different classes of programs

A decade of research* on the Legate runtime enables scalable implicit parallelism



Benchmarks with nearest neighbor communication



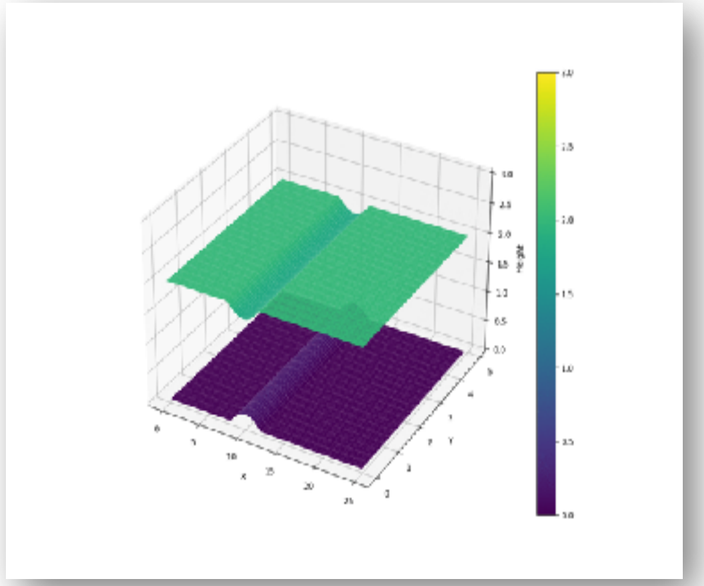
Benchmarks with logarithmic communication complexity

* Visibility Algorithms for Dynamic Dependence Analysis and Distributed Coherence, PPOPP 2023
Scaling Implicit Parallelism via Dynamic Control Replication, PPOPP 2021
Legion: Expressing Locality and Independence with Logical Regions, SC 2012

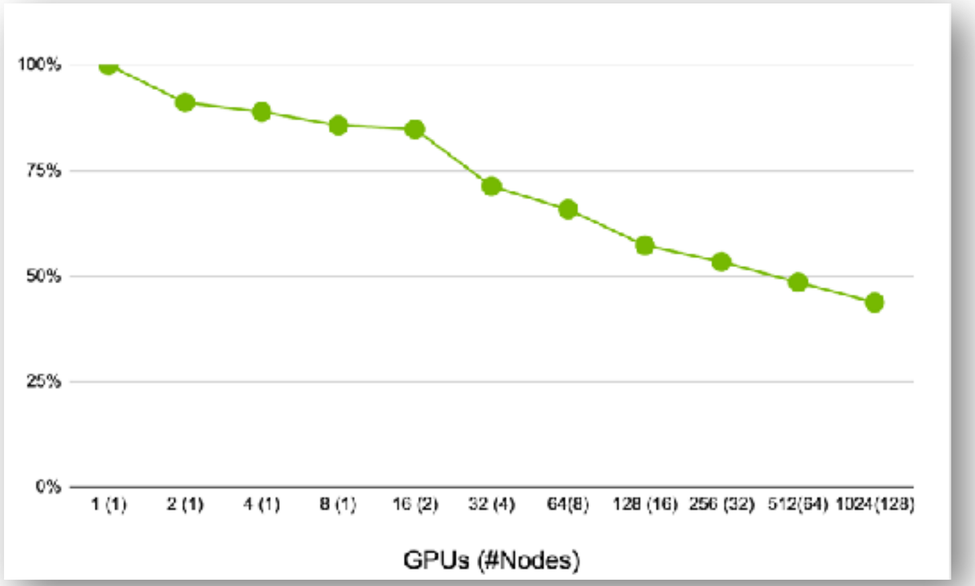
Out-of-the-Box Scaling for End-Users

TorchSWE: Porting an existing simulation to cuNumeric

- GPU accelerated solver for shallow water equations
- Original CuPy+MPI code is ported to cuNumeric by removing the MPI code
- cuNumeric allows domain scientists **with no MPI knowledge** to attain the simulation resolution (~20B points) historically restricted to only a few scientific groups



Topography (below) and water level (above)



Weak scaling performance (40M grid points / GPU)

Productive Development of Scalable, Composable Libraries

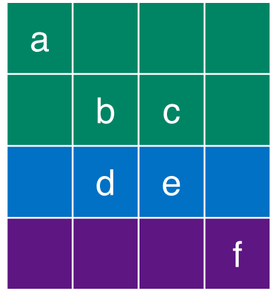
Legate Sparse: a SciPy-Sparse Implementation in Legate

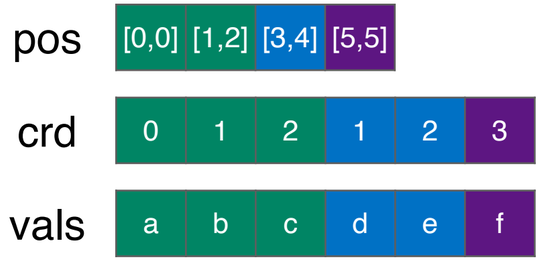
- ~35% of the APIs for 4 sparse matrix formats (CSR, CSC, COO, DIA) in 3 months
  - **Data-dependent partition support** played a key role in rapid development

```
# Create a banded diagonal matrix in CSR format
A = legate.sparse.diags(
    [1] * nnz_per_row,
    [x - (nnz_per_row // 2) for x in range(nnz_per_row)],
    shape=(n, n),
    format="csr",
)
# Create a dense vector
x = cunumeric.ones((n,))
# Perform SpMV
y = A.dot(x)
```

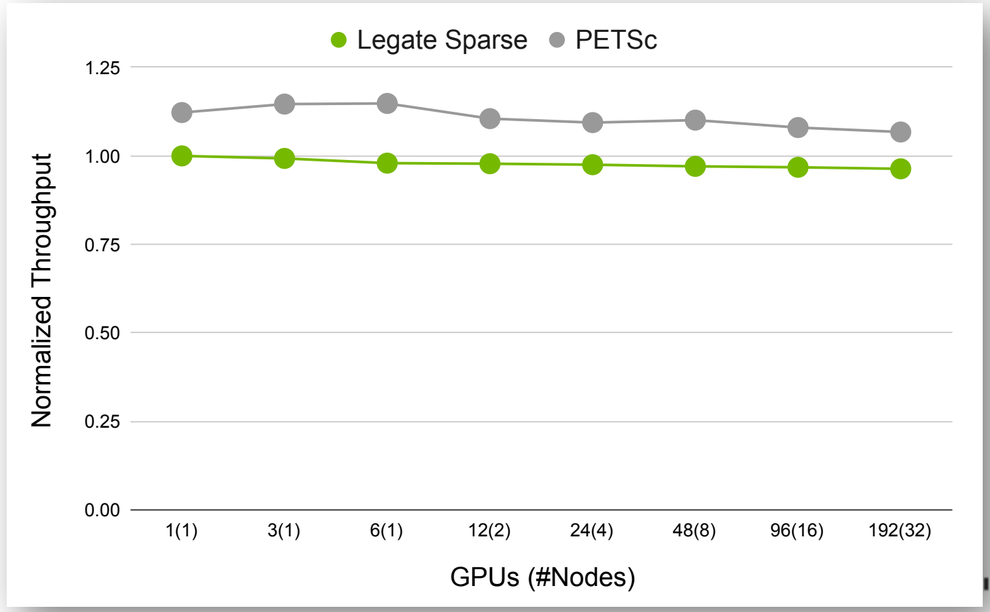
SpMV in Legate Sparse + cuNumeric

Legate Sparse can accept cuNumeric arrays





CSR matrix partitioned by row



Comparable performance with PETSc, a state-of-the-art MPI implementation

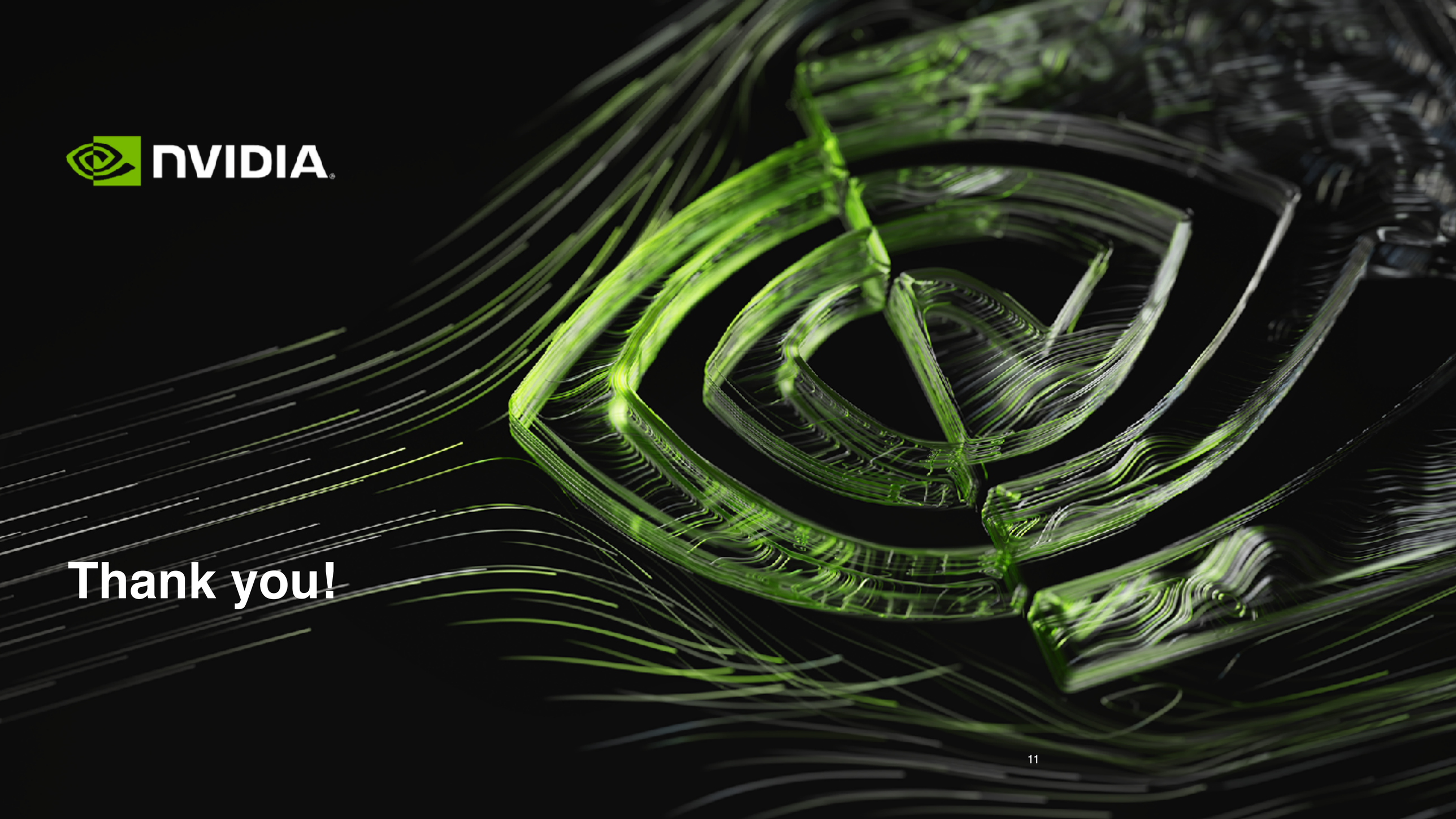
Join Us On Our Journey!

- Legate is a framework for developing scalable, composable software that end-users can “just use” with no expertise in distributed systems

- Please reach out to us on legate@nvidia.com if you
  - Have applications to scale and would like to take advantage of Legate’s ecosystem of libraries
  - Work on libraries that you want to make scalable and portable for distributed hardware

- Legate libraries can be found at https://github.com/nv-legate/

- Learn more:
  - Accelerating Python Applications with cuNumeric and Legate
  - cuNumeric and Legate web page



Thank you!