# Accelerating Sparse Applications

Fred Kjolstad
Assistant Professor, Stanford University

# Dense and Sparse Processing

## Sensors

# Dense and Sparse Processing

## Sensors

Images, Audio,
Signal Streams

## Dense Processing

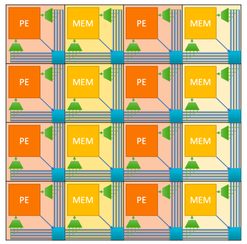
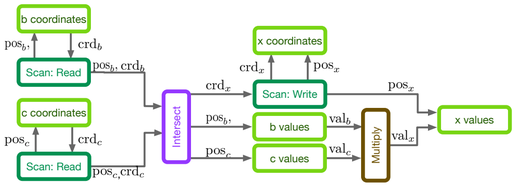



Image Processing,
DNN/CNN

## Data Centers

Databases,
Knowledge Graphs

## Sparse Processing

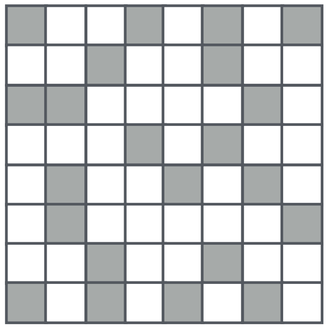



Data Processing,
Graph Algorithms and GNNs,
Sparse Tensor Factorizations

More abstract data

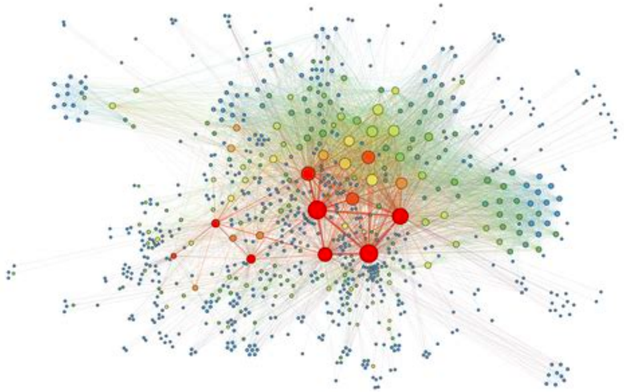
# How Sparse is Sparse?

## Fractional Sparsity



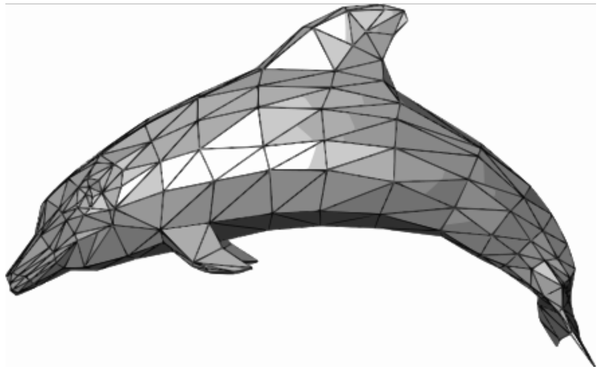
E.g., regularized DNN/CNN weight matrices

## Power-law Degrees, Small Diameter



E.g., social networks

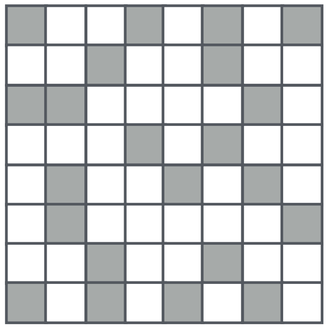
## Bounded Degrees, Large Diameter



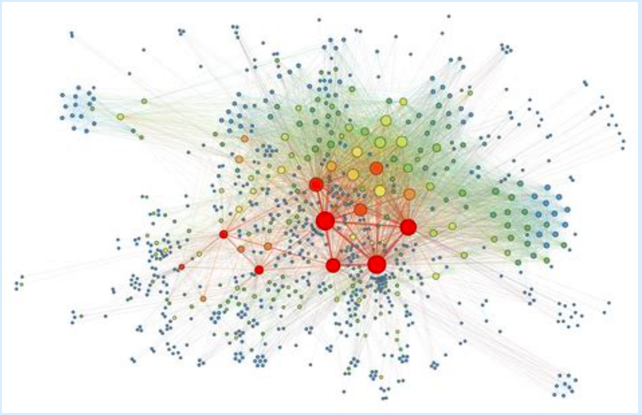
E.g., discretized physical objects, road networks

# How Sparse is Sparse?

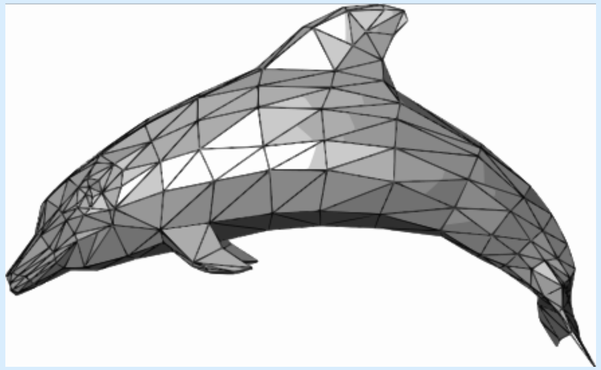
Fractional Sparsity



Power-law Degrees,
Small Diameter

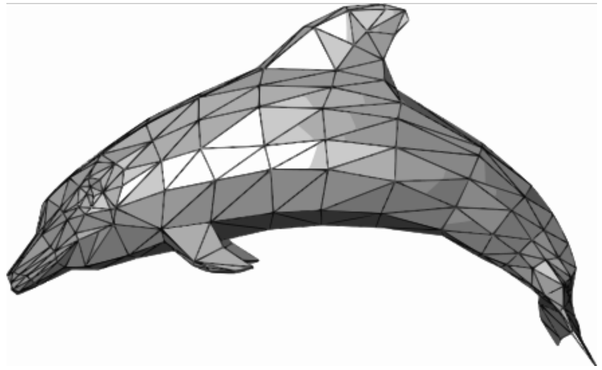


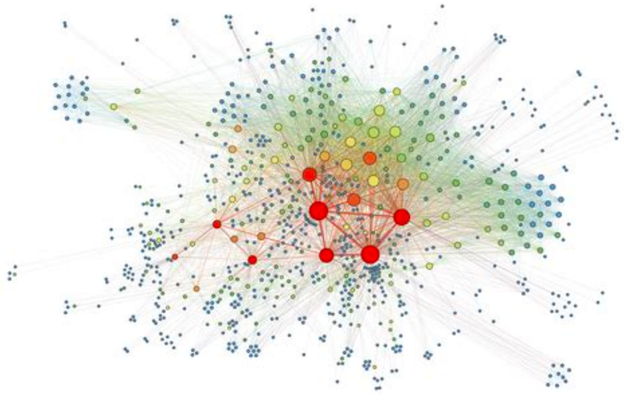
Bounded Degrees,
Large Diameter



Asymptotically sparse

# How Sparse is Sparse? Can be Asymptotic





Assume an average degree of 12 (e.g., 12 friends)

Each matrix row then has 12 nonzeros

At 10,000 rows: $\frac{12 \cdot 10{,}000}{10{,}000^2} = 0.12\,\%$ nonzeros

At 100,000 rows: $\frac{12 \cdot 100{,}000}{100{,}000^2} = 0.012\,\%$ nonzeros

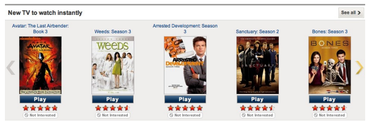
Matrix components: $O(n^2)$

Nonzeros: $O(n)$

Fraction of nonzeros: $O(1/n)$

# Sparse Tensor Algebra Applications

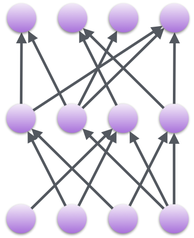
## Data Analytics



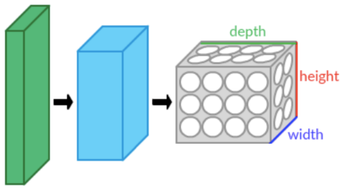

Movies

Social Networks

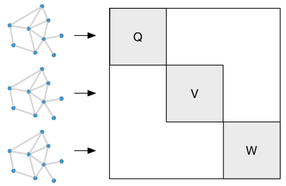
## Machine Learning



Sparse Networks



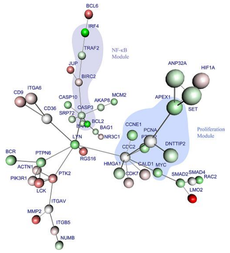

Sparse Convolutional Networks



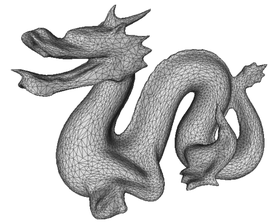

Graph Convolutional Network

## Science and Engineering

Robotics

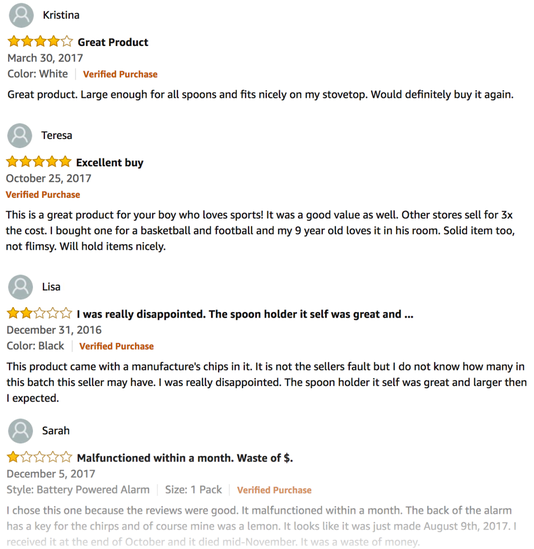


Computational Biology



Simulations

amazon

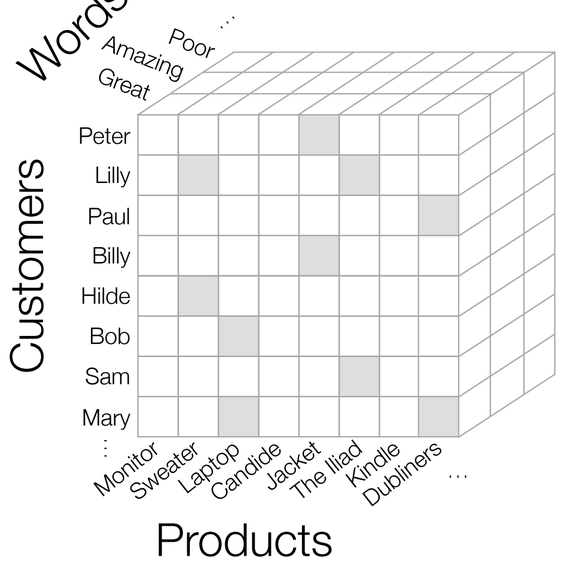


Kristina
**Great Product**
March 30, 2017
Color: White | Verified Purchase
Great product. Large enough for all spoons and fits nicely on my stovetop. Would definitely buy it again.

Teresa
**Excellent buy**
October 25, 2017
Verified Purchase
This is a great product for your boy who loves sports! It was a good value as well. Other stores sell for 3x the cost. I bought one for a basketball and football and my 9 year old loves it in his room. Solid item too, not flimsy. Will hold items nicely.

Lisa
**I was really disappointed. The spoon holder it self was great and ...**
December 31, 2016
Color: Black | Verified Purchase
This product came with a manufacture's chips in it. It is not the sellers fault but I do not know how many in this batch this seller may have. I was really disappointed. The spoon holder it self was great and larger then I expected.

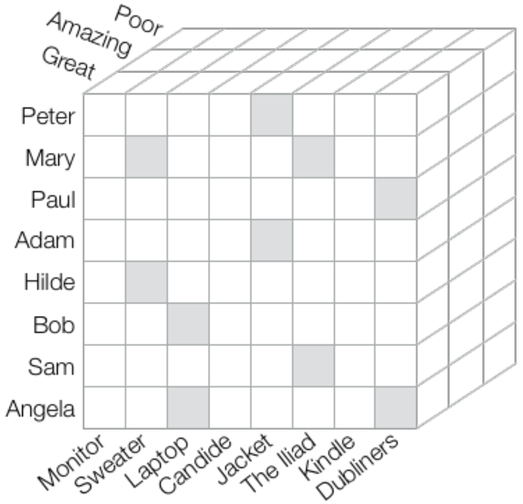
Sarah
**Malfunctioned within a month. Waste of $.**
December 5, 2017
Style: Battery Powered Alarm | Size: 1 Pack | Verified Purchase
I chose this one because the reviews were good. It malfunctioned within a month. The back of the alarm has a key for the chirps and of course mine was a lemon. It looks like it was just made August 9th, 2017. I received it at the end of October and it died mid-November. It was a waste of money.



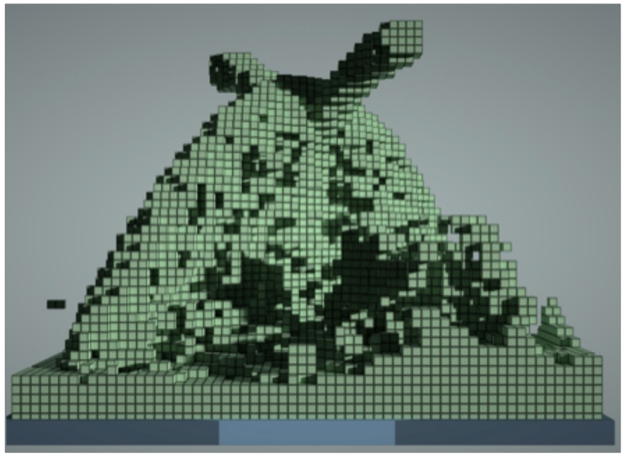

**Extremely sparse**
Dense storage: 107 Exabytes
Sparse storage: 13 Gigabytes

# Sparse Array Generalization

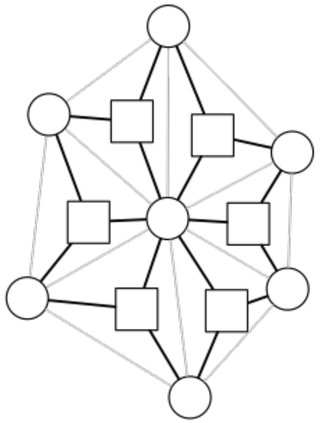
## Sparse Tensors




## Sparse Grids*



## Graphs



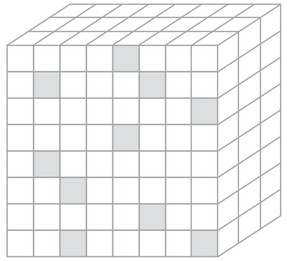
## Relations

| Names | City | Age |
|---|---|---|
| Peter | Boston | 54 |
| Mary | San Fransisco | 35 |
| Paul | New York | 23 |
| Adam | Seattle | 84 |
| Hilde | Boston | 19 |
| Bob | Chicago | 76 |
| Sam | Portland | 32 |
| Angela | Los Angeles | 62 |

*Hu et al. *Taichi: a language for high-performance computation on spatially sparse data structures,* 2019

# Mapping Data Models to Sparse Arrays

Tensors



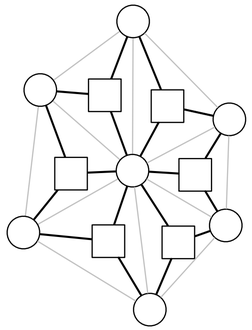
Modes map to dimensions,
nonzeros map to stored elements

Relations

| Names | City | Age |
|---|---|---|
| Peter | Boston | 54 |
| Mary | San Fransisco | 35 |
| Paul | New York | 23 |
| Adam | Seattle | 84 |
| Hilde | Boston | 19 |
| Bob | Chicago | 76 |
| Sam | Portland | 32 |
| Angela | Los Angeles | 62 |

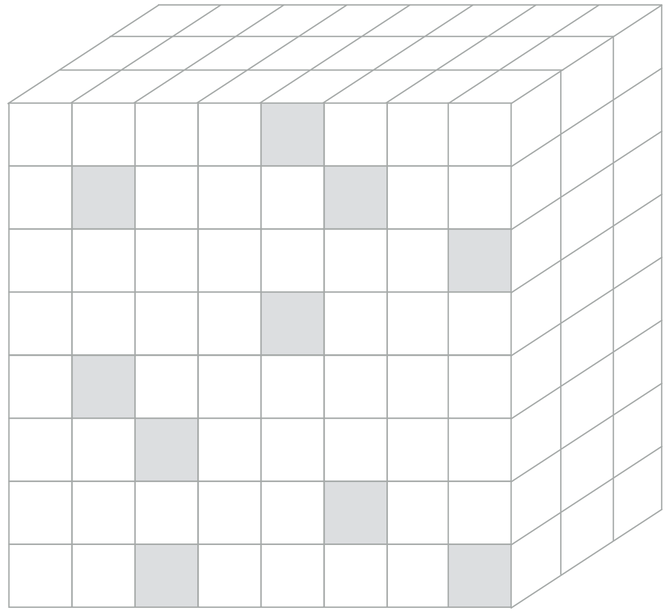
Columns map to dimensions,
rows map to stored elements

Graphs



Vertices map to dimensions,
edges map to stored elements

Sparse Array



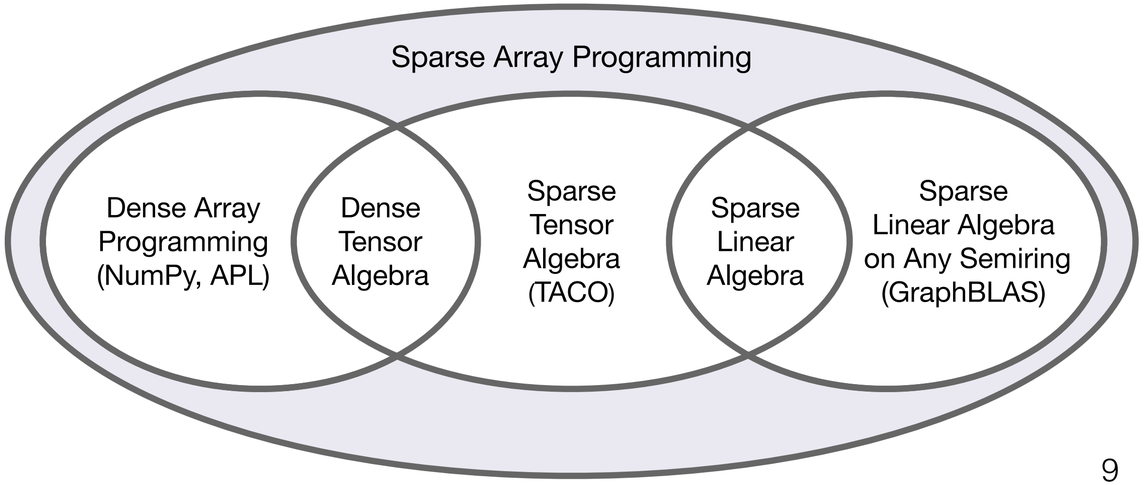
# SciPy Roadmap

1. Support for distributed arrays and GPU arrays
2. Performance improvements
3. Statistics enhancements
4. Support for more hardware platforms
5. Implement sparse arrays in addition to sparse matrices

https://docs.scipy.org/doc/scipy/reference/roadmap.html

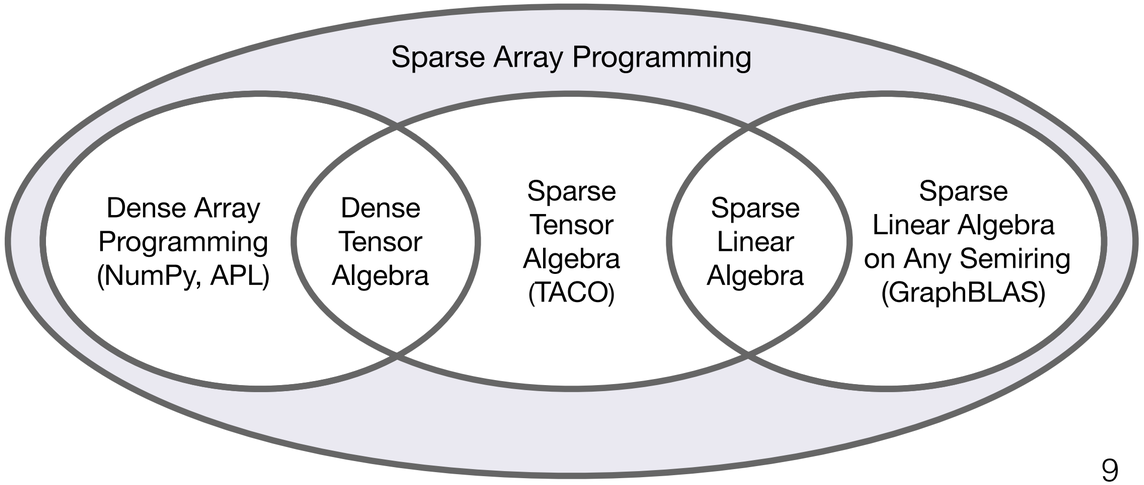
# Sparse Array Programming

| Paradigm | Supported Functions | | | Data Representation | | | | Slicing |
|---|---|---|---|---|---|---|---|---|
| | $(+, \times)$ | Any semiring $(\wedge, \vee), \ldots$ | Any `foo`, ... | Dense | Sparse: Zero fill | Sparse: Any fill | Any # of dims. | |
| Dense Array Programming (NumPy) | ✔ | ✔ | ✔ | ✔ | ✘ | ✘ | ✔ | ✔ |
| Dense Tensor Algebra | ✔ | ✘ | ✘ | ✔ | ✘ | ✘ | ✔ | ✔ |
| Sparse Tensor Algebra (TACO) | ✔ | ✘ | ✘ | ✔ | ✔ | ✘ | ✔ | ✘ |
| Sparse Linear Algebra | ✔ | ✘ | ✘ | ✔ | ✔ | ✘ | ✘ | ✘ |
| Sparse LA on Any Semiring (GraphBLAS) | ✔ | ✔ | ✘ | ✔ | ✔ | ✘ | ✘ | ✔ |
| Sparse Array Programming (This Work) | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |



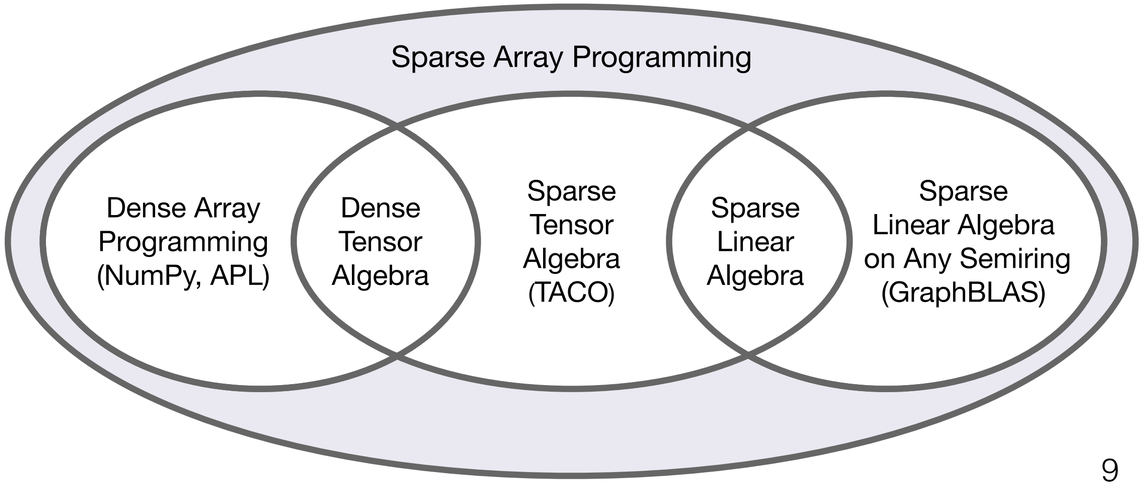
# Sparse Array Programming

| Paradigm | Supported Functions | | | Data Representation | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Any semiring | Any | | Sparse | | Any # | Slicing |
| | $(+, \times)$ | $(\wedge, \vee), \ldots$ | `foo`, ... | Dense | Zero fill | Any fill | of dims. | |
| Dense Array Programming (NumPy) | ✔ | ✔ | ✔ | ✔ | ✘ | ✘ | ✔ | ✔ |
| Dense Tensor Algebra | ✔ | ✘ | ✘ | ✔ | ✘ | ✘ | ✔ | ✔ |
| Sparse Tensor Algebra (TACO) | ✔ | ✘ | ✘ | ✔ | ✔ | ✘ | ✔ | ✘ |
| Sparse Linear Algebra | ✔ | ✘ | ✘ | ✔ | ✔ | ✘ | ✘ | ✘ |
| Sparse LA on Any Semiring (GraphBLAS) | ✔ | ✔ | ✘ | ✔ | ✔ | ✘ | ✘ | ✔ |
| Sparse Array Programming (This Work) | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |



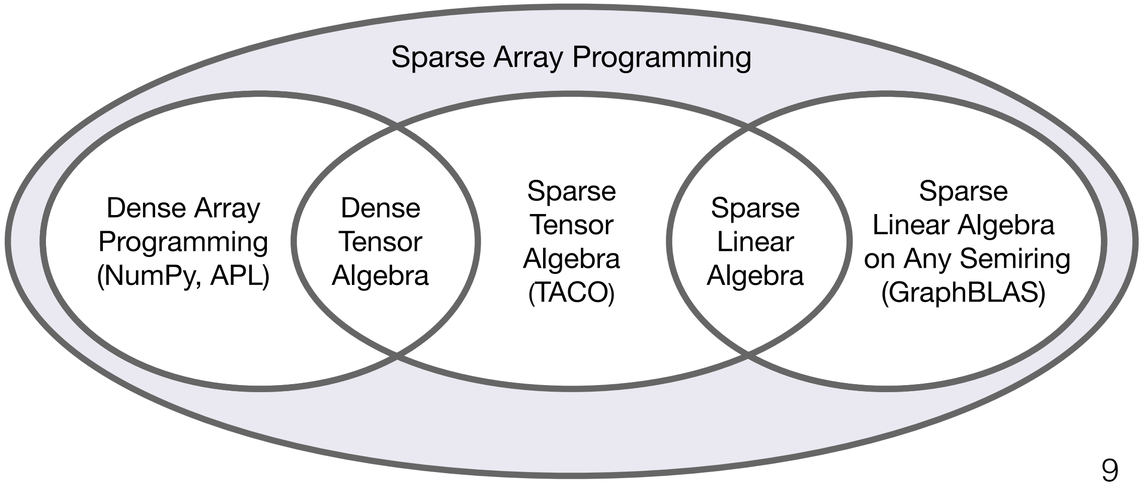
# Sparse Array Programming

| Paradigm | Supported Functions | | | Data Representation | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Any semiring | Any | | Sparse | | Any # | Slicing |
| | $(+, \times)$ | $(\wedge, \vee), \ldots$ | `foo`, ... | Dense | Zero fill | Any fill | of dims. | |
| Dense Array Programming (NumPy) | ✔ | ✔ | ✔ | ✔ | ✘ | ✘ | ✔ | ✔ |
| Dense Tensor Algebra | ✔ | ✘ | ✘ | ✔ | ✘ | ✘ | ✔ | ✔ |
| Sparse Tensor Algebra (TACO) | ✔ | ✘ | ✘ | ✔ | ✔ | ✘ | ✔ | ✘ |
| Sparse Linear Algebra | ✔ | ✘ | ✘ | ✔ | ✔ | ✘ | ✘ | ✘ |
| Sparse LA on Any Semiring (GraphBLAS) | ✔ | ✔ | ✘ | ✔ | ✔ | ✘ | ✘ | ✔ |
| Sparse Array Programming (This Work) | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |



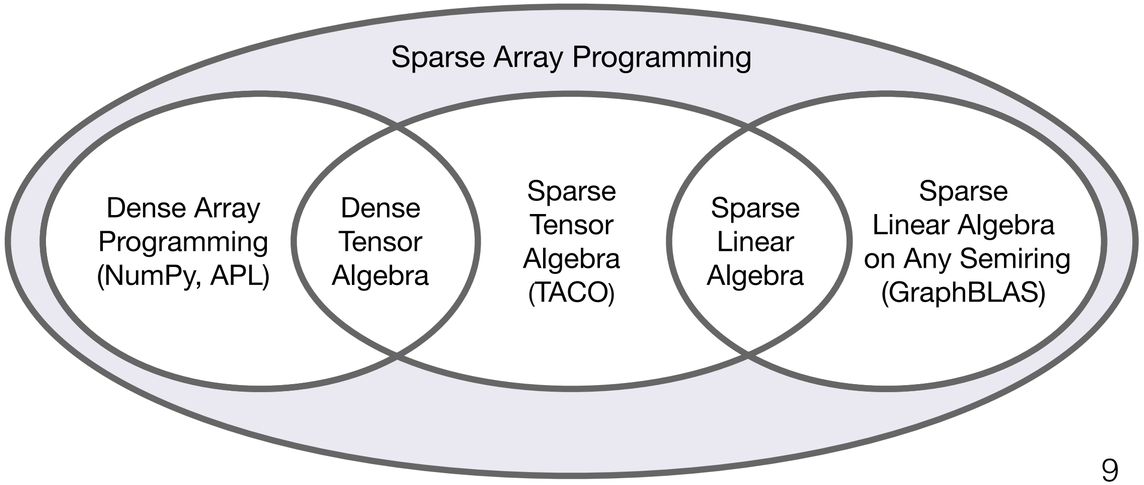
# Sparse Array Programming

| Paradigm | Supported Functions | | | Data Representation | | | | Slicing |
|---|---|---|---|---|---|---|---|---|
| | $(+, \times)$ | Any semiring $(\wedge, \vee), \ldots$ | Any `foo`, ... | Dense | Sparse: Zero fill | Sparse: Any fill | Any # of dims. | |
| Dense Array Programming (NumPy) | ✔ | ✔ | ✔ | ✔ | ✘ | ✘ | ✔ | ✔ |
| Dense Tensor Algebra | ✔ | ✘ | ✘ | ✔ | ✘ | ✘ | ✔ | ✔ |
| Sparse Tensor Algebra (TACO) | ✔ | ✘ | ✘ | ✔ | ✔ | ✘ | ✔ | ✘ |
| Sparse Linear Algebra | ✔ | ✘ | ✘ | ✔ | ✔ | ✘ | ✘ | ✘ |
| Sparse LA on Any Semiring (GraphBLAS) | ✔ | ✔ | ✘ | ✔ | ✔ | ✘ | ✘ | ✔ |
| Sparse Array Programming (This Work) | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

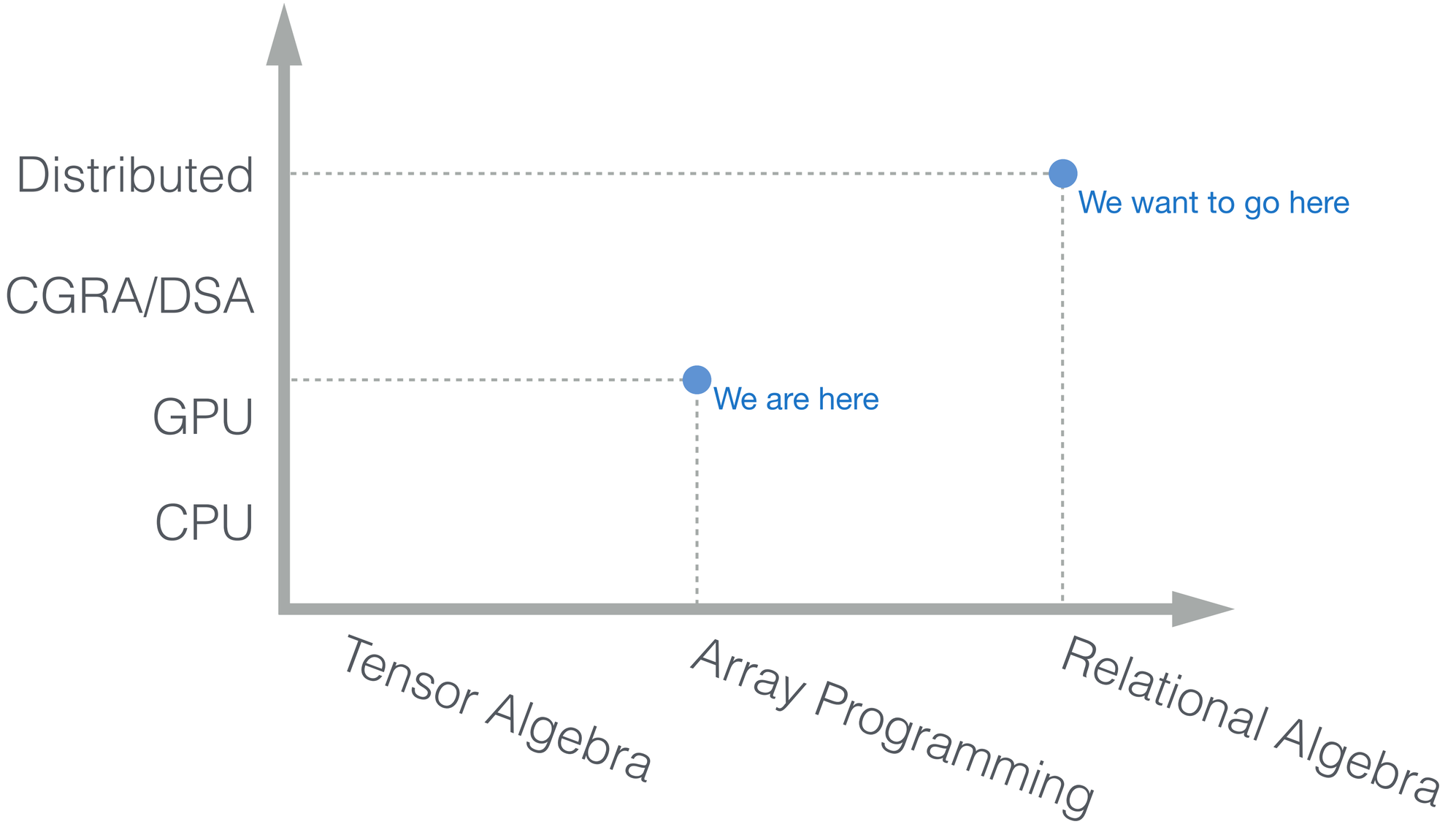


# Sparse Array Programming

| Paradigm | Supported Functions | | | Data Representation | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Any semiring | Any | | Sparse | | Any # | Slicing |
| | $(+, \times)$ | $(\wedge, \vee), \ldots$ | `foo`, ... | Dense | Zero fill | Any fill | of dims. | |
| Dense Array Programming (NumPy) | ✔ | ✔ | ✔ | ✔ | ✘ | ✘ | ✔ | ✔ |
| Dense Tensor Algebra | ✔ | ✘ | ✘ | ✔ | ✘ | ✘ | ✔ | ✔ |
| Sparse Tensor Algebra (TACO) | ✔ | ✘ | ✘ | ✔ | ✔ | ✘ | ✔ | ✘ |
| Sparse Linear Algebra | ✔ | ✘ | ✘ | ✔ | ✔ | ✘ | ✘ | ✘ |
| Sparse LA on Any Semiring (GraphBLAS) | ✔ | ✔ | ✘ | ✔ | ✔ | ✘ | ✘ | ✔ |
| Sparse Array Programming (This Work) | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |



# Sparse Roadmap



# Why you need a compiler

$$a = Bc$$
$$a = Bc + a$$
$$a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a$$
$$a = B^T c \quad A = \alpha B \quad a = B(c + d)$$
$$a = B^T c + d \quad A = B + C + D \quad A = BC$$
$$A = B \odot C \quad a = b \odot c \; A = 0 \quad A = B \odot (CD)$$
$$A = BCd \quad A = B^T \quad a = B^T Bc$$
$$a = b + c \quad A = B \quad K = A^T CA$$
$$A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij}$$
$$A_{lj} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \quad A_{ij} = \sum_{k} B_{ijk} c_k$$
$$A_{ijk} = \sum_{l} B_{ikl} C_{lj} \quad A_{ik} = \sum_{j} B_{ijk} c_j$$
$$A_{jk} = \sum_{i} B_{ijk} c_i \quad A_{ijl} = \sum_{k} B_{ikl} C_{kj}$$
$$\tau = \sum_i z_i (\sum_j z_j \theta_{ij})(\sum_k z_k \theta_{ik})$$
$$C = \sum_{ijkl} M_{ij} P_{jk} \overline{M_{lk}} \, \overline{P_{il}}$$
$$a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} \overline{M_{nm}} P_{no} \overline{M_{po}} \, \overline{P_{ip}}$$

×

Dense Matrix

CSR DCSR BCSR

COO ELLPACK CSB

Blocked COO CSC

DIA Blocked DIA DCSC

Sparse vector Hash Maps

Coordinates

CSF Dense Tensors

Blocked Tensors

Linked Lists Database

Compression Schemes

Cloud Storage

×

CPU

GPUs TPUs

FPGA

Sparse Tensor Hardware

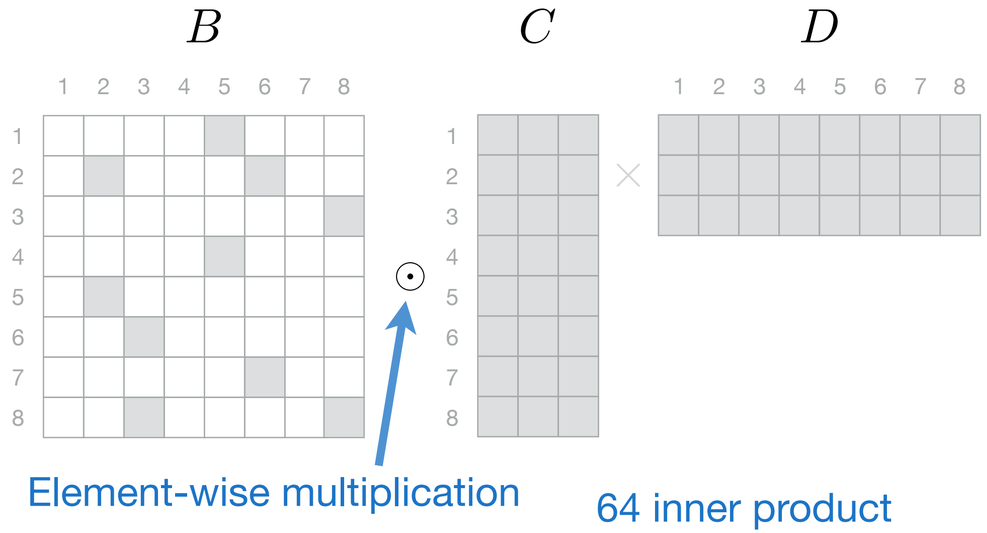
Cloud Computers

Supercomputers

# Why you need a compiler

$$a = Bc$$
$$a = Bc + a$$
$$a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a$$
$$a = B^T c \quad A = \alpha B \quad a = B(c + d)$$
$$a = B^T c + d \quad A = B + C + D \quad A = BC$$
$$A = B \odot C \quad a = b \odot c \; A = 0 \quad A = B \odot (CD)$$
$$A = BCd \quad A = B^T \quad a = B^T Bc$$
$$a = b + c \quad A = B \quad K = A^T C A$$
$$A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij}$$
$$A_{lj} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \quad A_{ij} = \sum_{k} B_{ijk} c_k$$
$$A_{ijk} = \sum_{l} B_{ikl} C_{lj} \quad A_{ik} = \sum_{j} B_{ijk} c_j$$
$$A_{jk} = \sum_{i} B_{ijk} c_i \quad A_{ijl} = \sum_{k} B_{ikl} C_{kj}$$
$$\tau = \sum_i z_i (\sum_j z_j \theta_{ij})(\sum_k z_k \theta_{ik})$$
$$C = \sum_{ijkl} M_{ij} P_{jk} \overline{M_{lk}} \, \overline{P_{il}}$$
$$a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} \overline{M_{nm}} P_{no} \overline{M_{po}} \, \overline{P_{ip}}$$

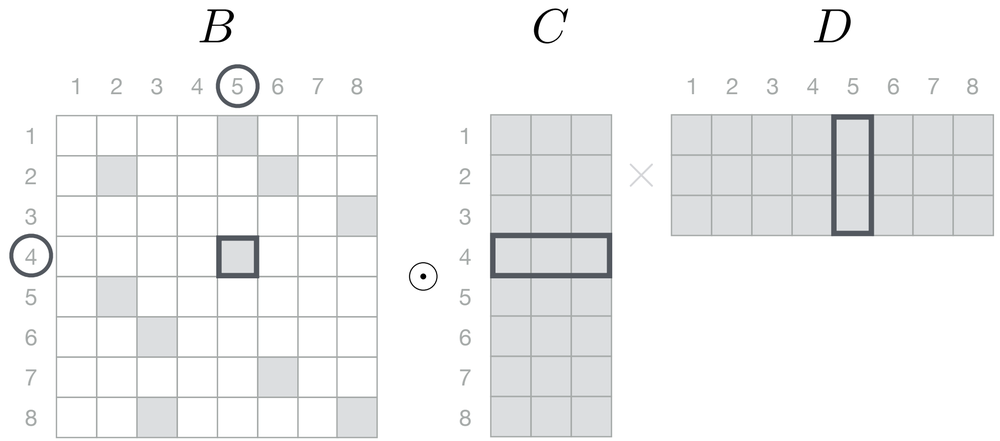
## Sampled Dense-Dense Matrix Multiplication (SDDMM)



# Why you need a compiler

$$a = Bc$$
$$a = Bc + a$$
$$a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a$$
$$a = B^T c \quad A = \alpha B \quad a = B(c + d)$$
$$a = B^T c + d \quad A = B + C + D \quad A = BC$$
$$A = B \odot C \quad a = b \odot c \; A = 0 \quad A = B \odot (CD)$$
$$A = BCd \quad A = B^T \quad a = B^T Bc$$
$$a = b + c \quad A = B \quad K = A^T C A$$
$$A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij}$$
$$A_{lj} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \quad A_{ij} = \sum_{k} B_{ijk} c_k$$
$$A_{ijk} = \sum_{l} B_{ikl} C_{lj} \quad A_{ik} = \sum_{j} B_{ijk} c_j$$
$$A_{jk} = \sum_{i} B_{ijk} c_i \quad A_{ijl} = \sum_{k} B_{ikl} C_{kj}$$
$$\tau = \sum_i z_i (\sum_j z_j \theta_{ij})(\sum_k z_k \theta_{ik})$$
$$C = \sum_{ijkl} M_{ij} P_{jk} \overline{M_{lk}} \overline{P_{il}}$$
$$a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} \overline{M_{nm}} P_{no} \overline{M_{po}} \overline{P_{ip}}$$

## Sampled Dense-Dense Matrix Multiplication (SDDMM)

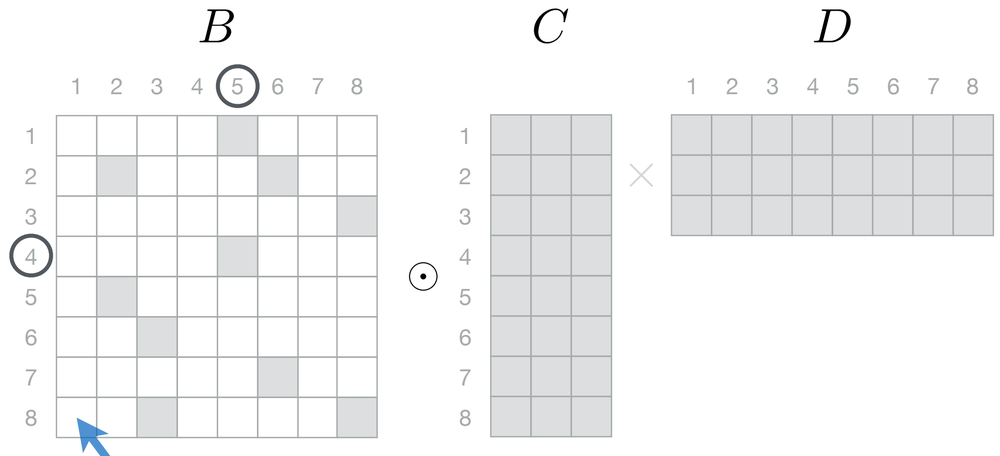



64 inner product

# Why you need a compiler

$$a = Bc$$
$$a = Bc + a$$
$$a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a$$
$$a = B^T c \quad A = \alpha B \quad a = B(c + d)$$
$$a = B^T c + d \quad A = B + C + D \quad A = BC$$
$$A = B \odot C \quad a = b \odot c \; A = 0 \quad A = B \odot (CD)$$
$$A = BCd \quad A = B^T \quad a = B^T Bc$$
$$a = b + c \quad A = B \quad K = A^T C A$$
$$A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij}$$
$$A_{lj} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \quad A_{ij} = \sum_{k} B_{ijk} c_k$$
$$A_{ijk} = \sum_{l} B_{ikl} C_{lj} \quad A_{ik} = \sum_{j} B_{ijk} c_j$$
$$A_{jk} = \sum_{i} B_{ijk} c_i \quad A_{ijl} = \sum_{k} B_{ikl} C_{kj}$$
$$\tau = \sum_i z_i (\sum_j z_j \theta_{ij})(\sum_k z_k \theta_{ik})$$
$$C = \sum_{ijkl} M_{ij} P_{jk} \overline{M_{lk}} \, \overline{P_{il}}$$
$$a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} \overline{M_{nm}} P_{no} \overline{M_{po}} \, \overline{P_{ip}}$$

## Sampled Dense-Dense Matrix Multiplication (SDDMM)



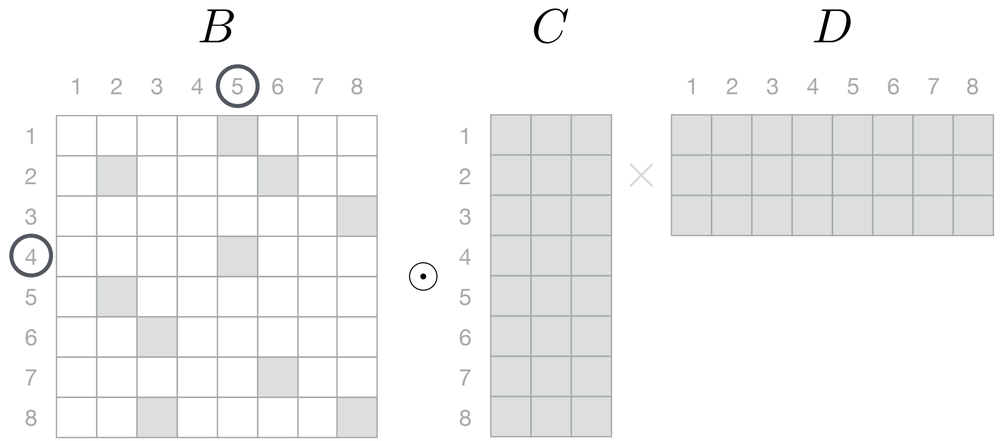

This dot product need not be computed

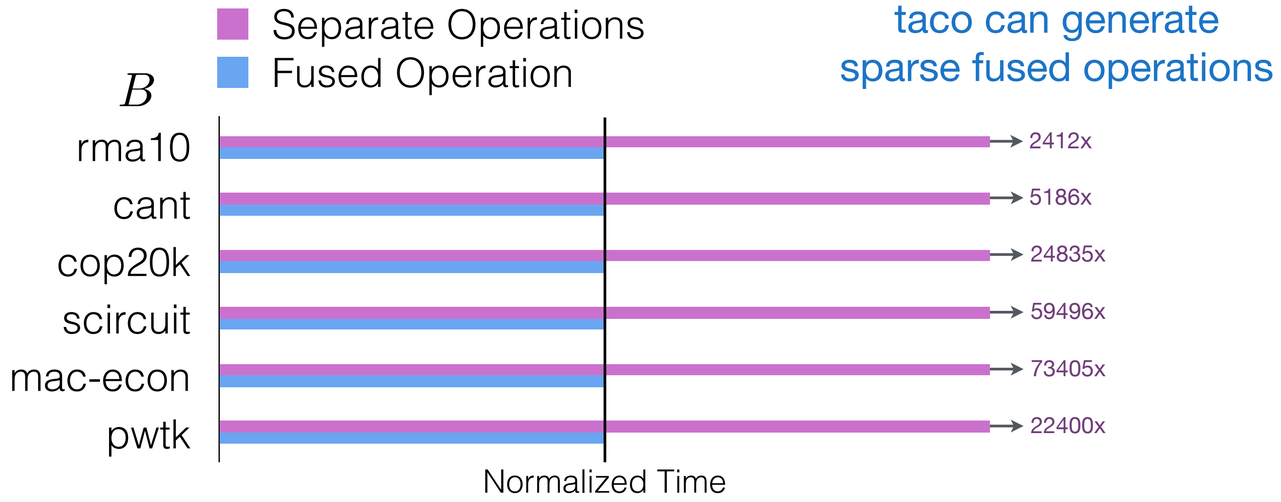
~~64 inner product~~
10 inner product

# Why you need a compiler

$$a = Bc$$
$$a = Bc + a$$
$$a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a$$
$$a = B^T c \quad A = \alpha B \quad a = B(c + d)$$
$$a = B^T c + d \quad A = B + C + D \quad A = BC$$
$$A = B \odot C \quad a = b \odot c \; A = 0 \quad A = B \odot (CD)$$
$$A = BCd \quad A = B^T \quad a = B^T Bc$$
$$a = b + c \quad A = B \quad K = A^T C A$$
$$A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij}$$
$$A_{lj} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \quad A_{ij} = \sum_{k} B_{ijk} c_k$$
$$A_{ijk} = \sum_{l} B_{ikl} C_{lj} \quad A_{ik} = \sum_{j} B_{ijk} c_j$$
$$A_{jk} = \sum_{i} B_{ijk} c_i \quad A_{ijl} = \sum_{k} B_{ikl} C_{kj}$$
$$\tau = \sum_{i} z_i (\sum_{j} z_j \theta_{ij})(\sum_{k} z_k \theta_{ik})$$
$$C = \sum_{ijkl} M_{ij} P_{jk} \overline{M_{lk}} \, \overline{P_{il}}$$
$$a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} \overline{M_{nm}} P_{no} \overline{M_{po}} \, \overline{P_{ip}}$$

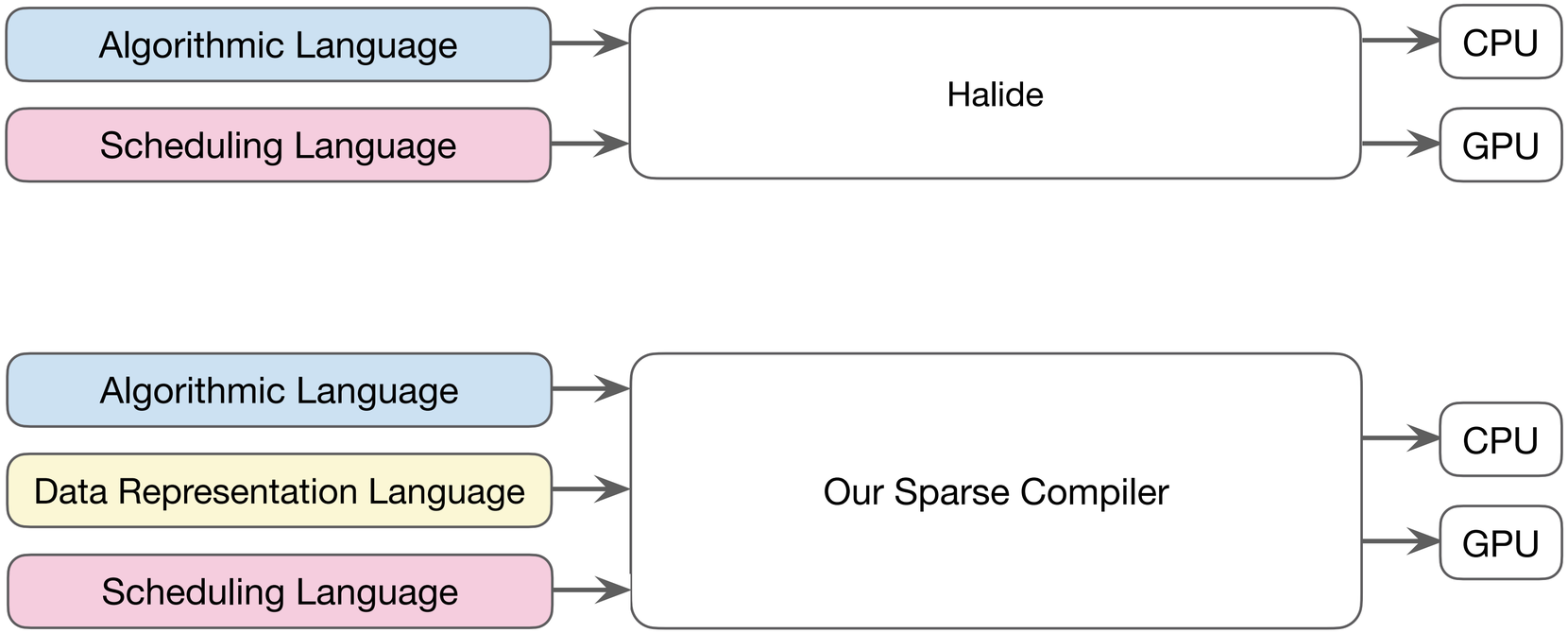
## Sampled Dense-Dense Matrix Multiplication (SDDMM)



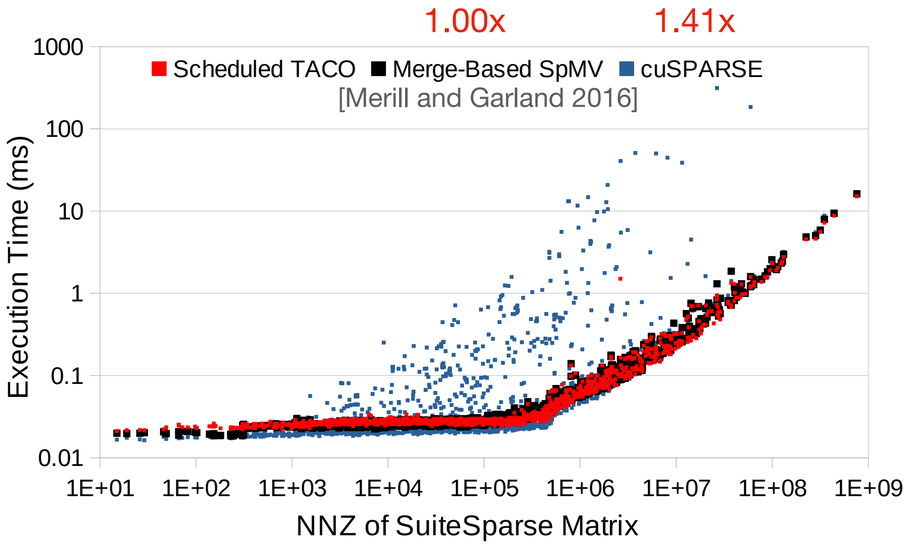

taco can generate
sparse fused operations

# Separation of Algorithm, Data Representation, and Schedule

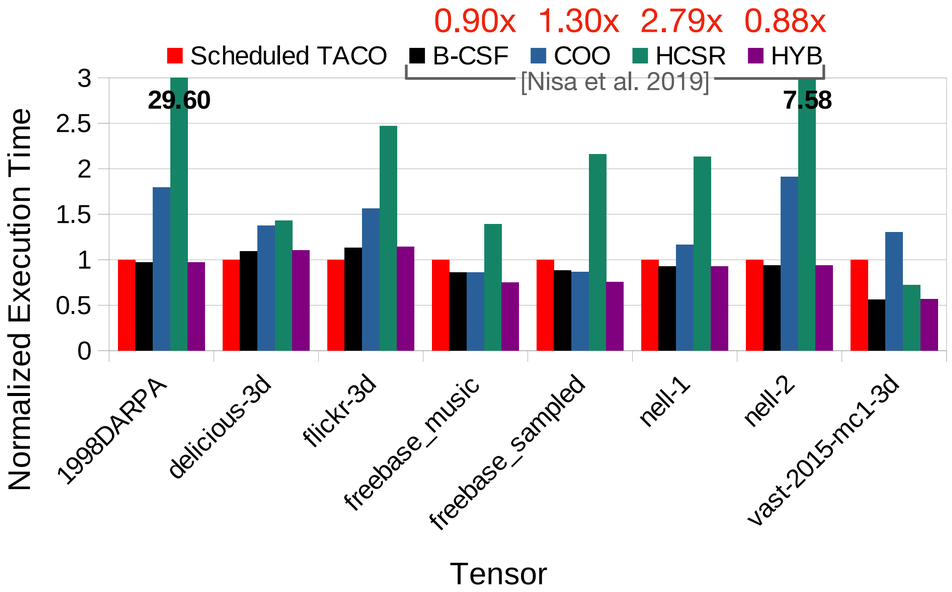


# Sparse Compilation Performance on CPUs and GPUs

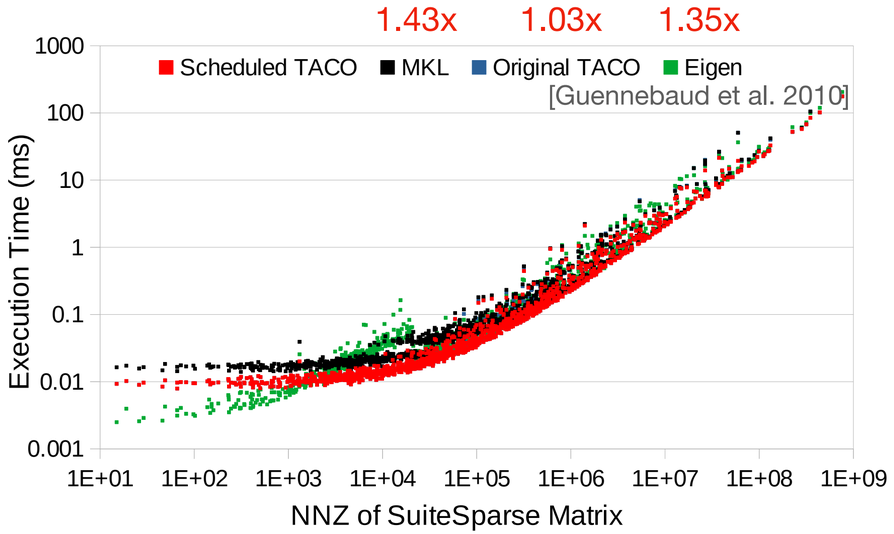
**GPU SpMV (NVIDIA V100)**



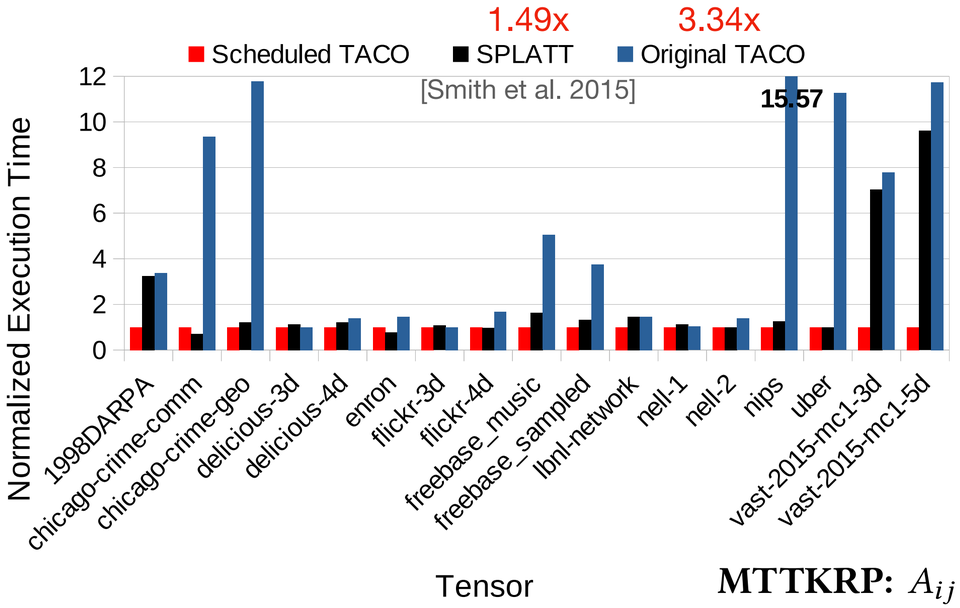

**GPU MTTKRP (NVIDIA V100)**




**CPU SpMV (12 cores)**




**CPU MTTKRP (12 cores)**




**MTTKRP:** $A_{ij} = \sum_{kl} B_{ikl} C_{kj} D_{lj}$

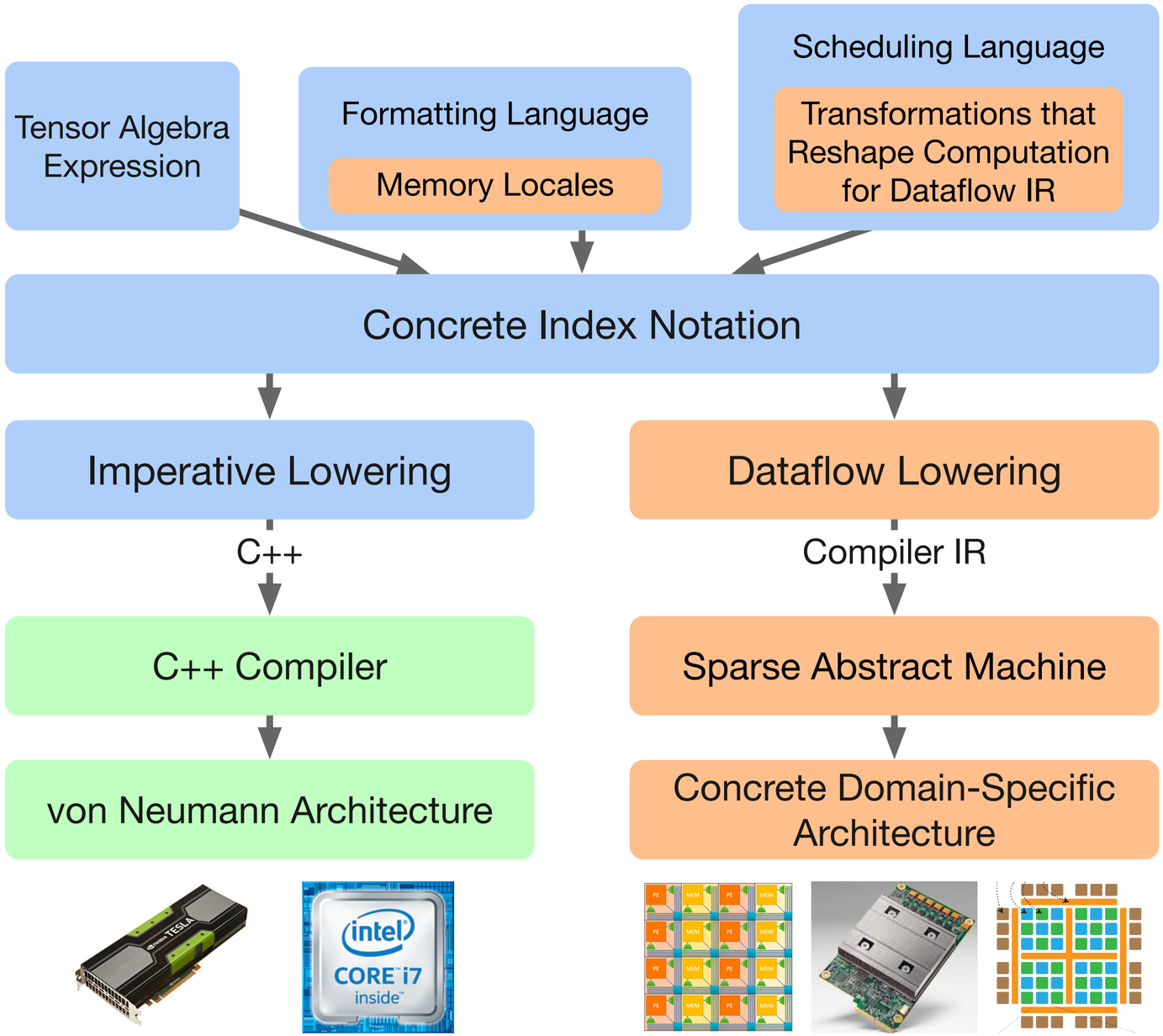

14

# Sparse Tensor Algebra Dataflow Compiler

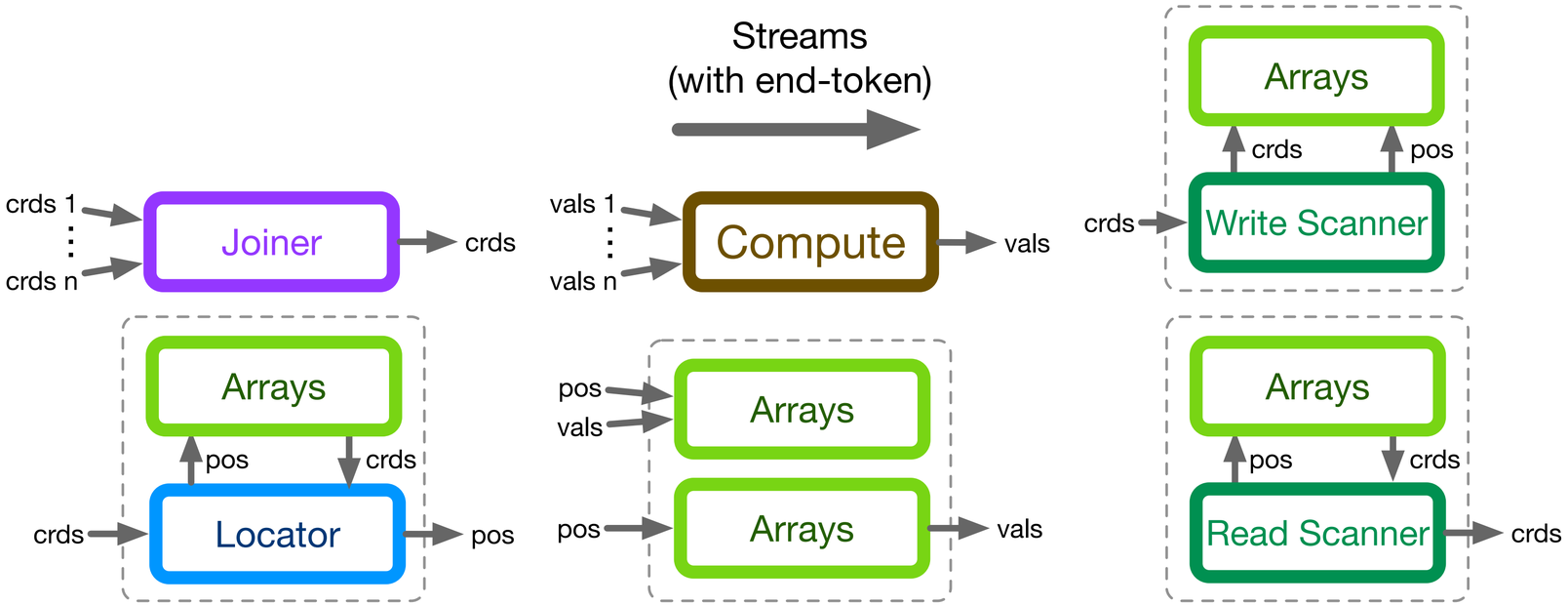
Imperative

Dataflow

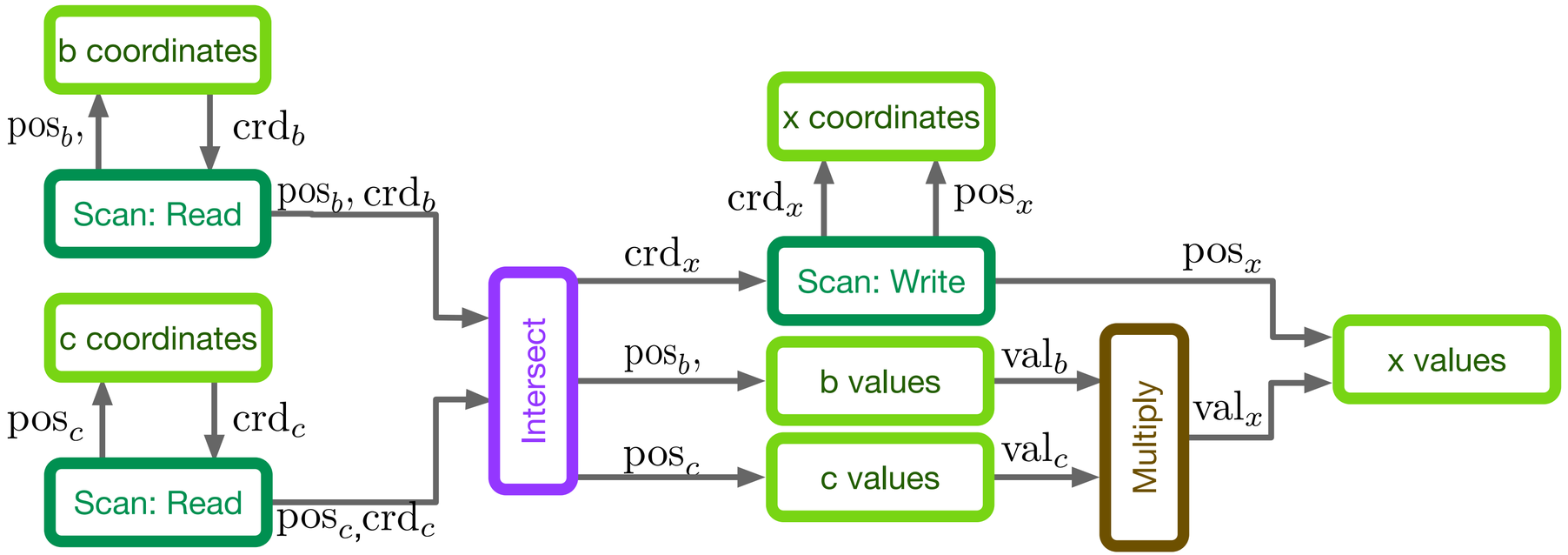
Other



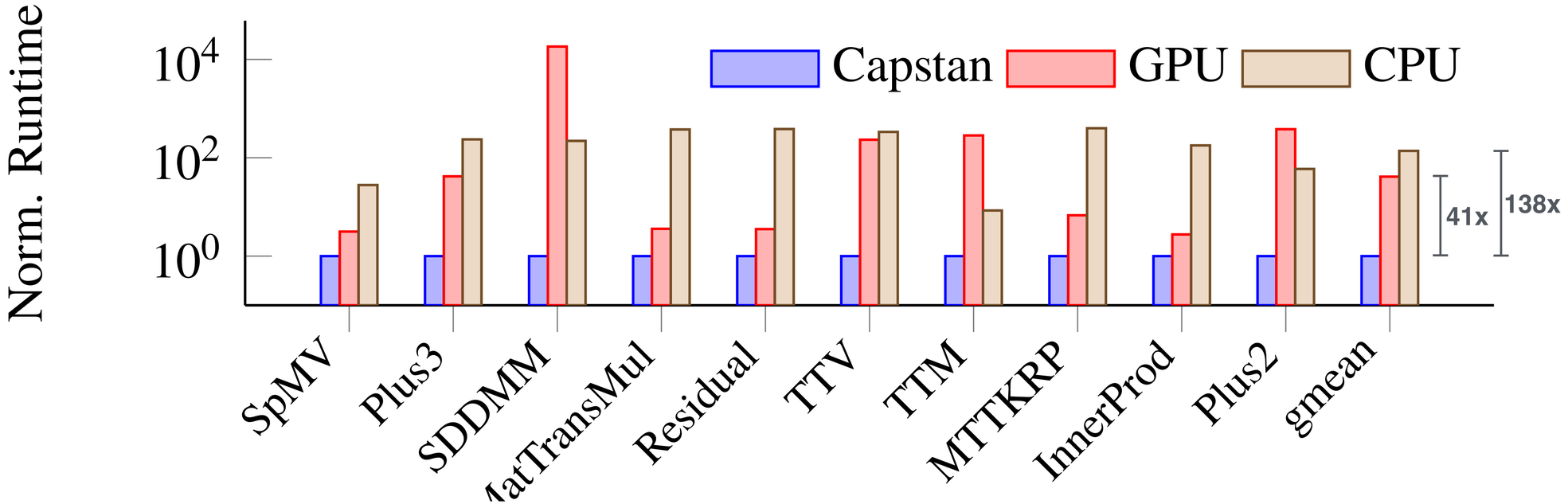
# Sparse Abstract Machine (IR)



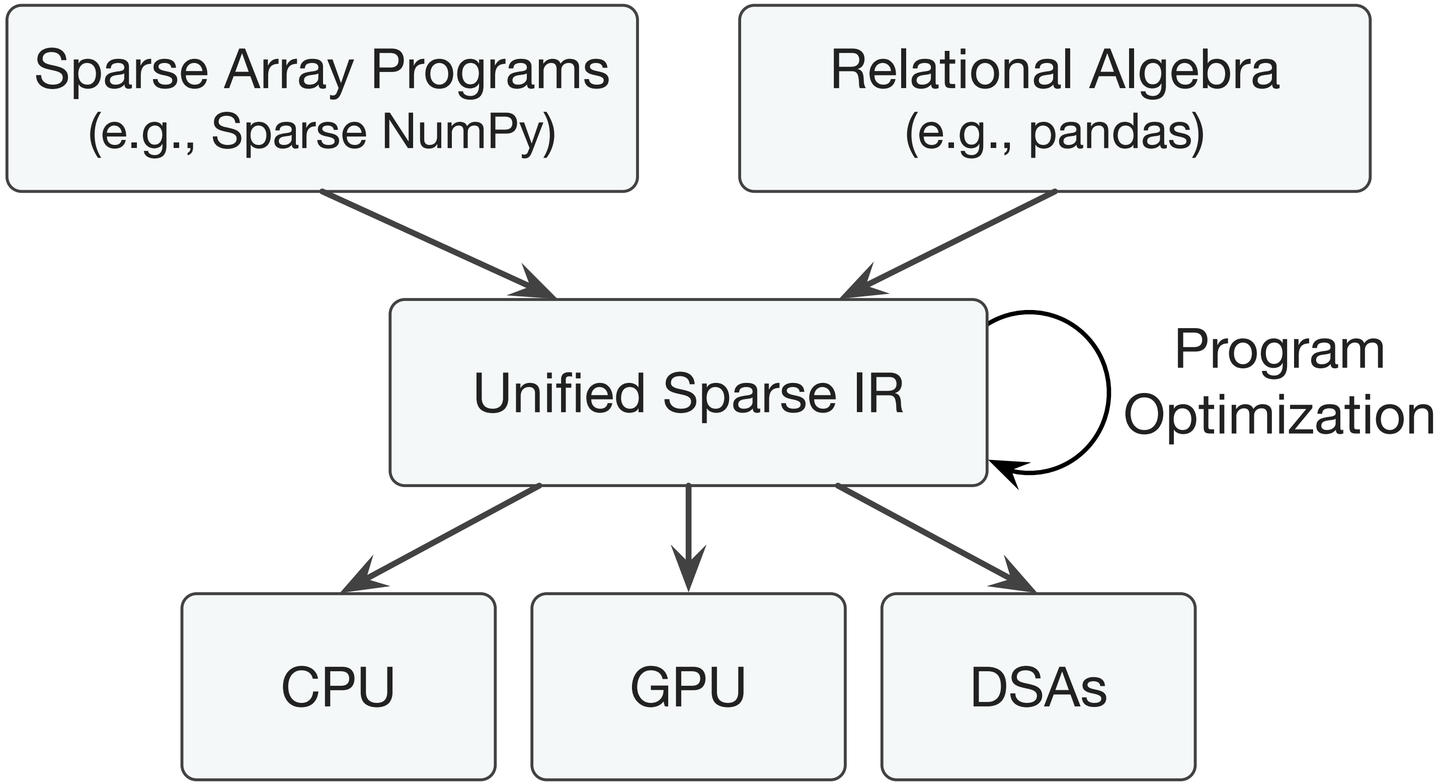
# Sparse Abstract Machine Example



# Initial Performance Results Targeting the Capstan Architecture



# Two Data Models, One Compiler, Many Targets



# Two Data Models

## Relations

| City | City | Distance |
|---|---|---|
| Los Angeles | New York | 2799 |
| New York | Boston | 220 |
| San Fransisco | New York | 2903 |
| Chicago | Seattle | 2043 |
| Boston | Los Angeles | 2983 |
| Seattle | Boston | 3045 |
| Chicago | New York | 806 |

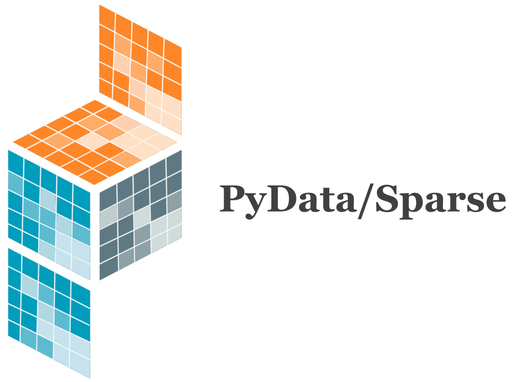
## Tensors

| | Los Angeles | New York | Boston | San Fransisco | Chicago | Seattle |
|---|---|---|---|---|---|---|
| Los Angeles | | 2799 | 2983 | | | |
| New York | 2799 | | 220 | 2903 | 806 | |
| Boston | 2983 | 220 | | | | 3045 |
| San Fransisco | | 2903 | | | | |
| Chicago | | 806 | | | | 2043 |
| Seattle | | | 3045 | | 2043 | |

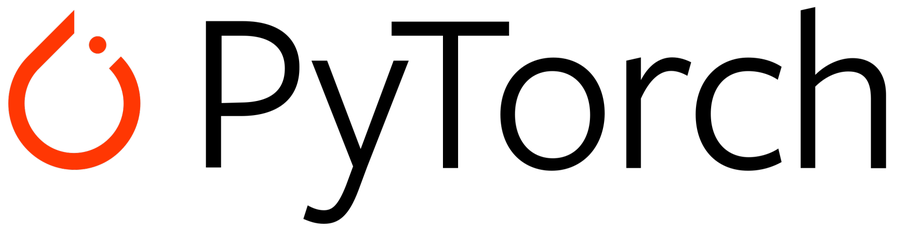
Convenient for Data Processing

Convenient for Computation

# User-Facing Library Ecosystem, e.g., in Python



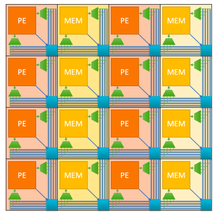
# Target Different Processor Mixes



+



+




## Distributed Machine