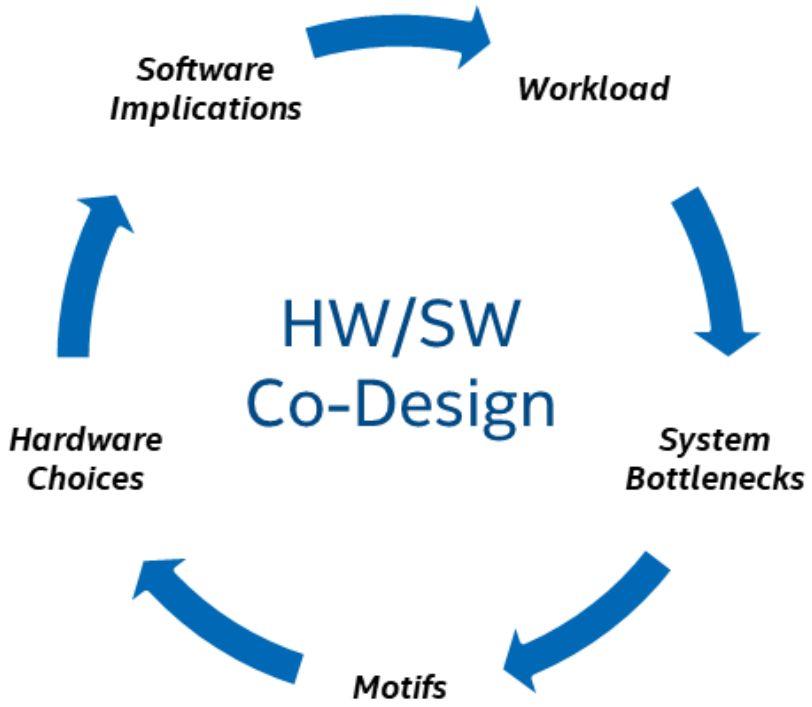
ALTO: Mode-Agnostic Linearized Storage for High-Performance Sparse Tensor Computations

Ahmed E. Helal,* Jan Laukemann,* Fabio Checconi,* Jesmin Jahan Tithi,*
Teresa Ranadive,† Fabrizio Petrini,* Jee Choi ‡

Intel labs,* Laboratory for Physical Sciences (LPS),† University of Oregon‡

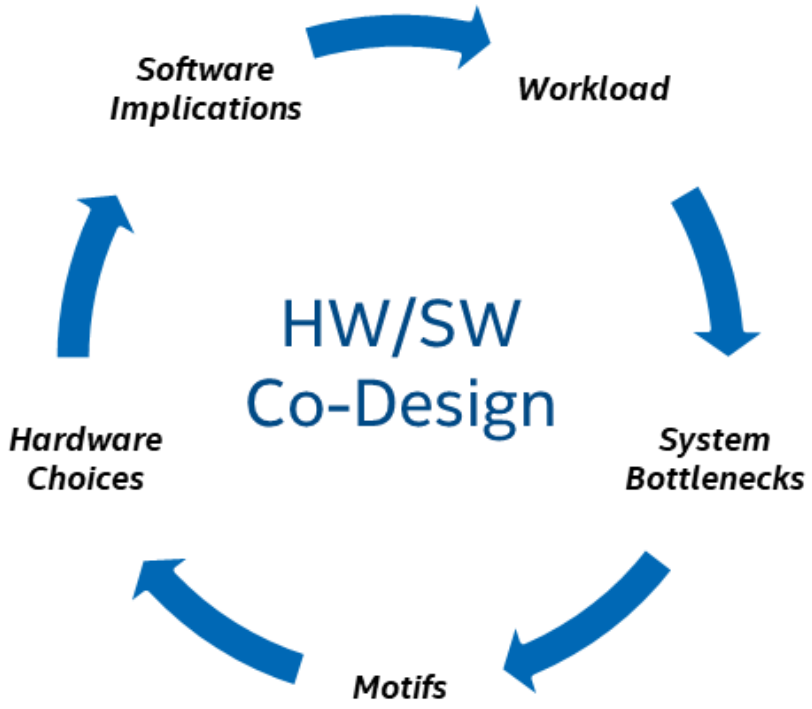
Intel PCL

Our modus operandi:



Intel PCL

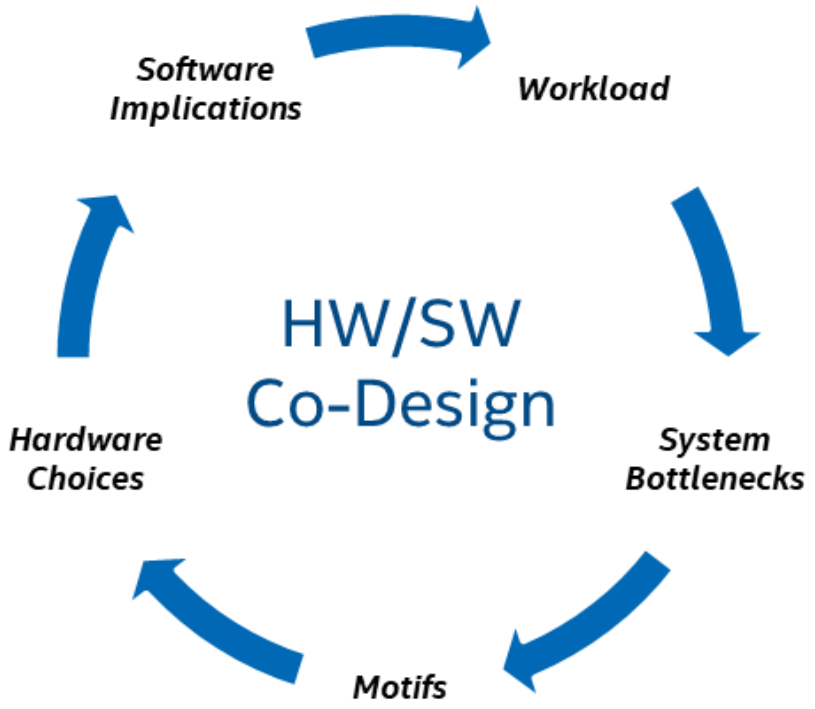
Our modus operandi:



- **Application-driven** research to improve **efficiency** and **performance** of Intel's architectures and tools

Intel PCL

Our modus operandi:



- **Application-driven** research to improve **efficiency** and **performance** of Intel's architectures and tools
- Target **domains**: machine learning (ML), high-performance computing (HPC), and data analytics

Outline

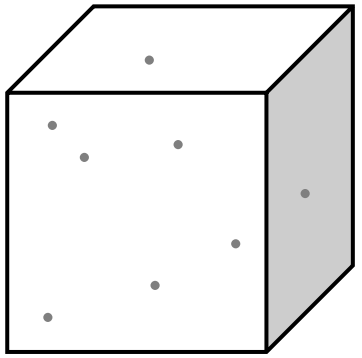
- Background & Motivation
  - State-of-the-art sparse tensor formats
- ALTO storage format
- Results & performance evaluation
- Conclusion

Motivation

- Many important application domains produce and manipulate massive amounts of **high-dimensional sparse data**

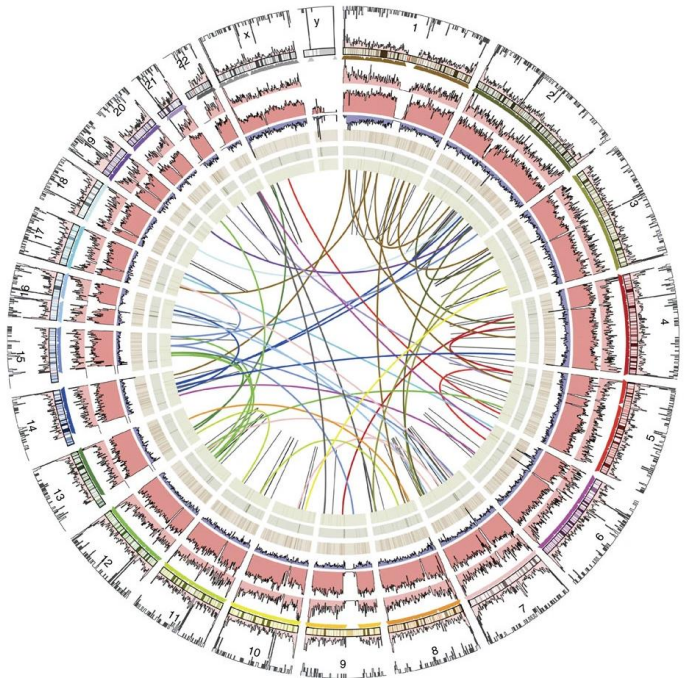
Motivation

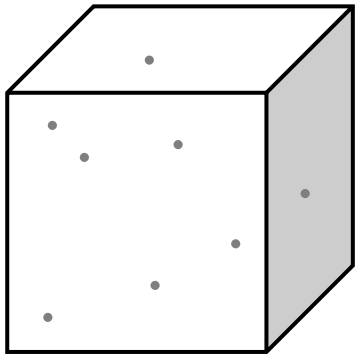
- Many important application domains produce and manipulate massive amounts of **high-dimensional sparse data**



Motivation

- Many important application domains produce and manipulate massive amounts of **high-dimensional sparse data**

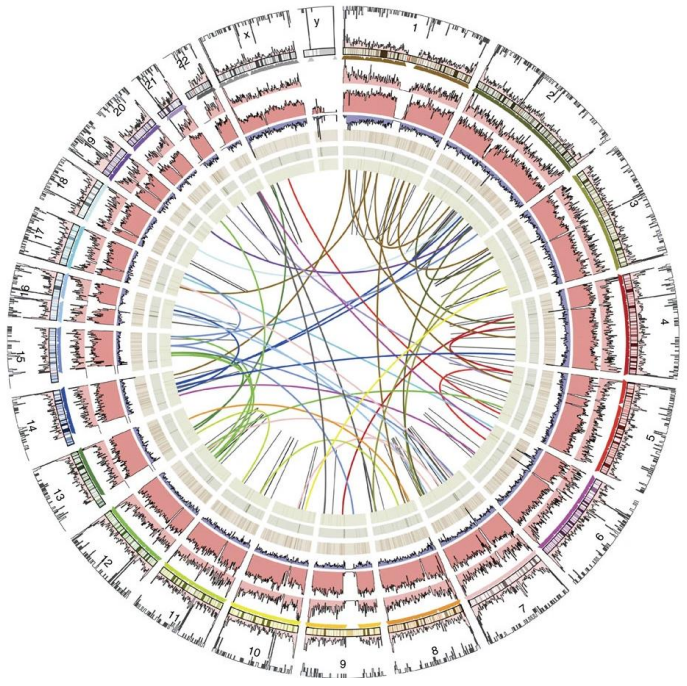


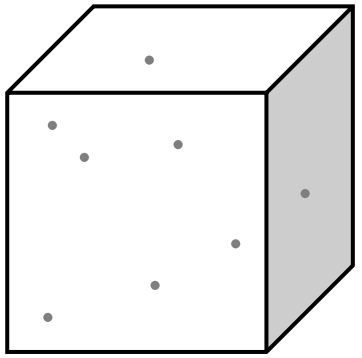


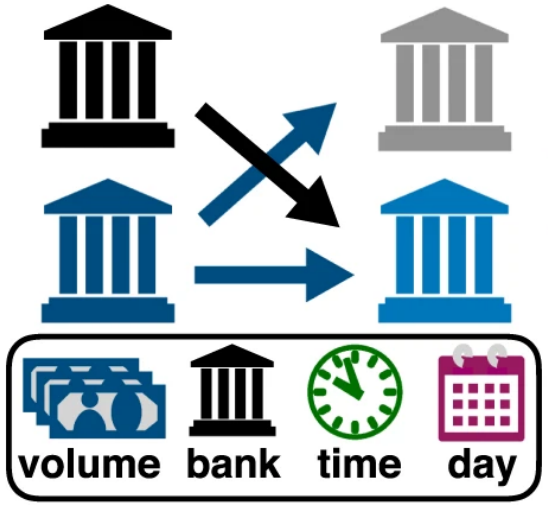
Source: Nature Communications and Scientific Reports, Wikimedia Commons

Motivation

- Many important application domains produce and manipulate massive amounts of **high-dimensional sparse data**



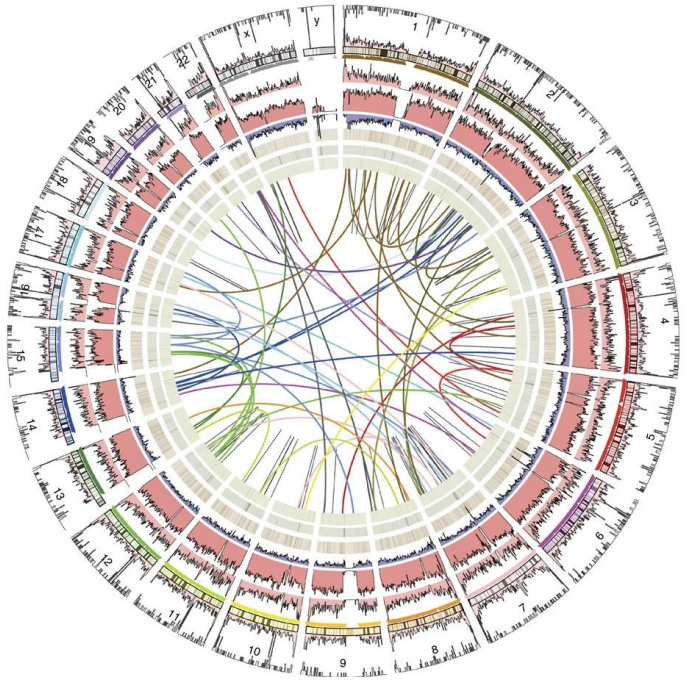


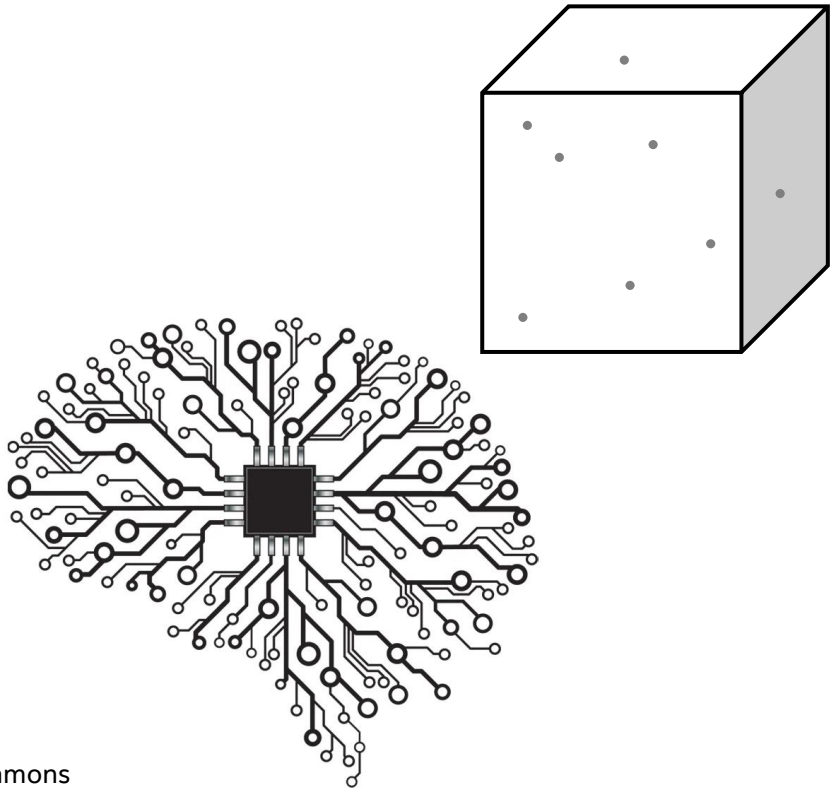


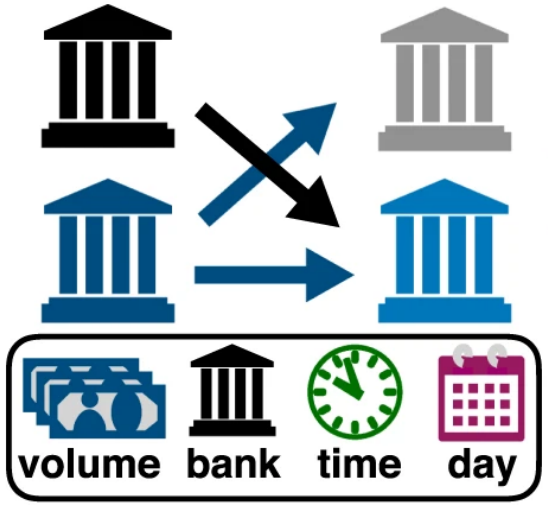
Source: Nature Communications and Scientific Reports, Wikimedia Commons

Motivation

- Many important application domains produce and manipulate massive amounts of **high-dimensional sparse data**



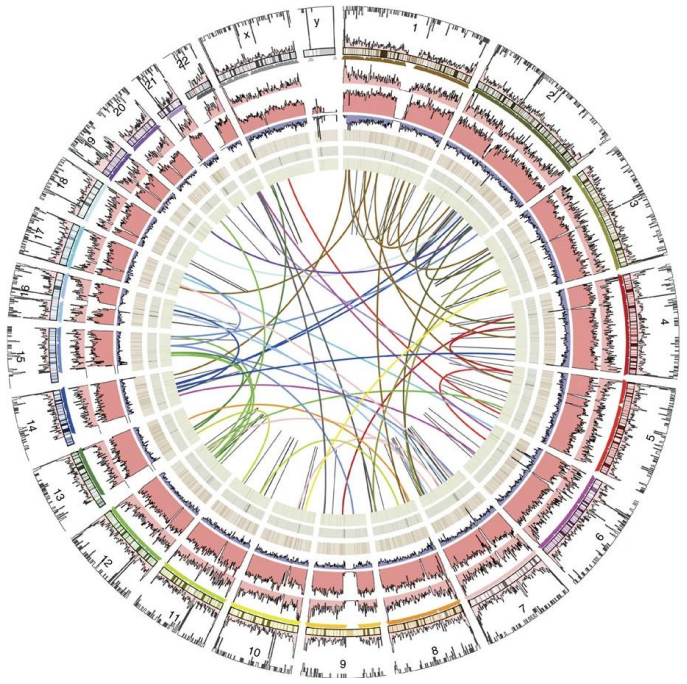


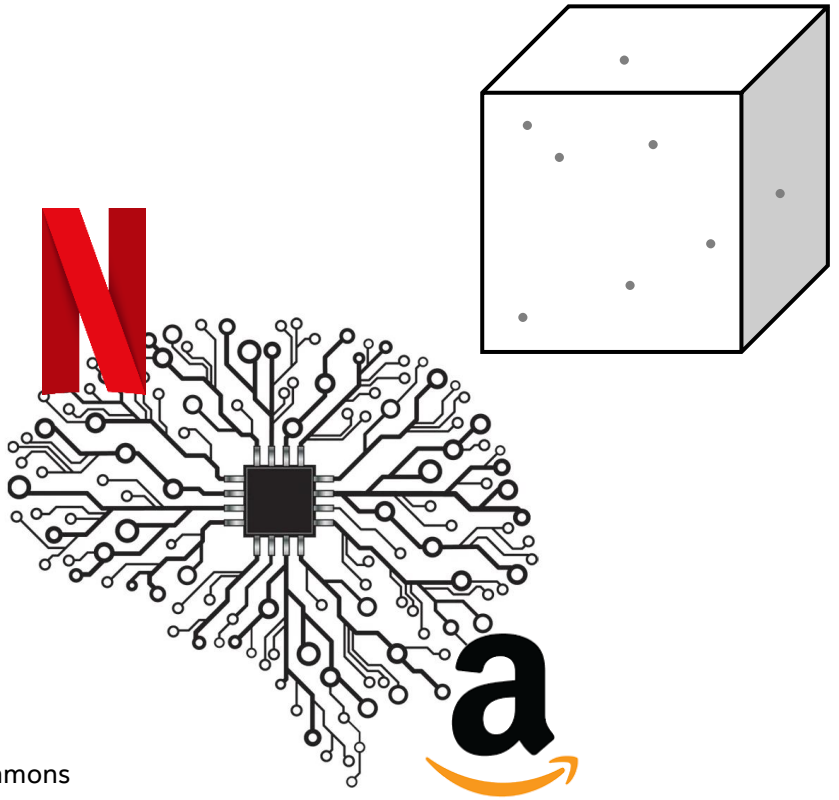


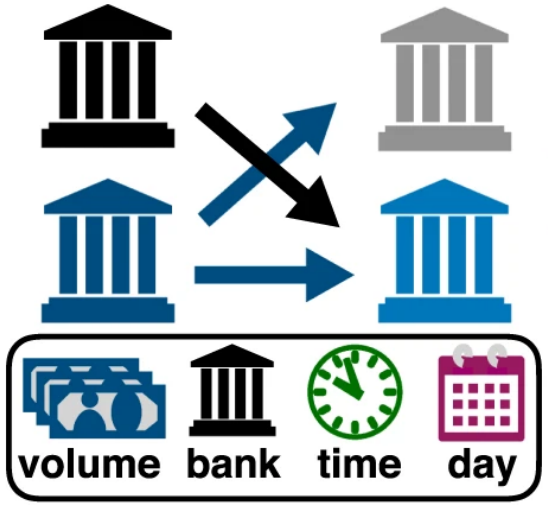
Source: Nature Communications and Scientific Reports, Wikimedia Commons

Motivation

- Many important application domains produce and manipulate massive amounts of **high-dimensional sparse data**



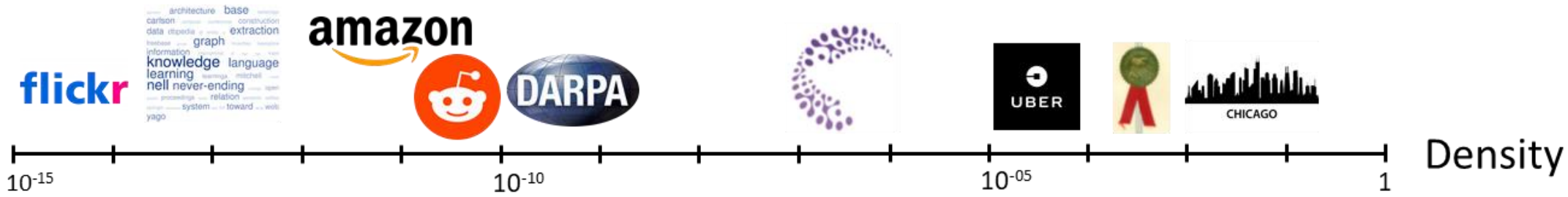




Source: Nature Communications and Scientific Reports, Wikimedia Commons

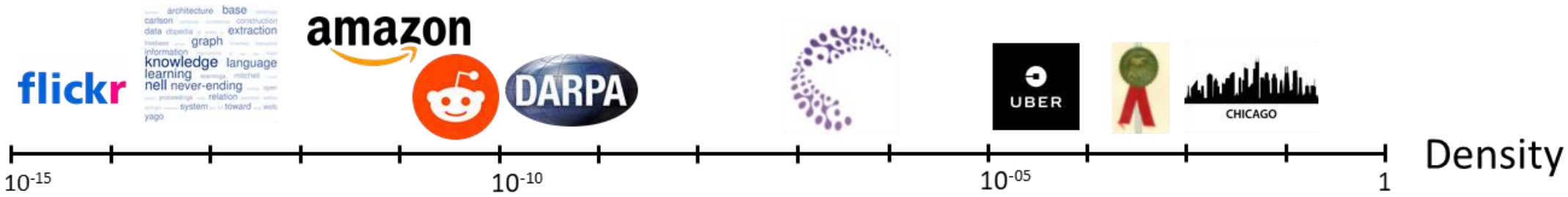
Motivation

- High-dimensional datasets exhibit diverse shapes and data distributions as well as unstructured sparsity patterns



Motivation

- High-dimensional datasets exhibit diverse shapes and data distributions as well as unstructured sparsity patterns
- Challenging to efficiently encode and represent



Motivation

- Store, group, and organize nonzero elements in sparse tensors is a **fundamental problem**
- Goals:

Motivation

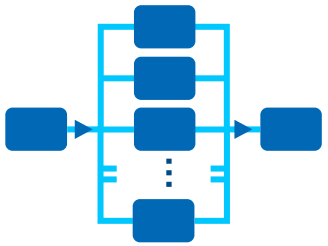
- Store, group, and organize nonzero elements in sparse tensors is a **fundamental problem**
- Goals:
  - ❖ reduce tensor **storage** 

Motivation

- Store, group, and organize nonzero elements in sparse tensors is a **fundamental problem**
- Goals:
  - reduce tensor **storage** 
  - improve **data locality** 

Motivation

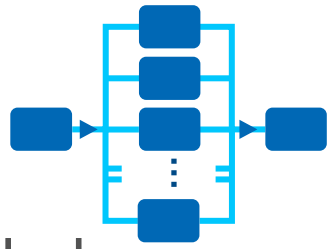
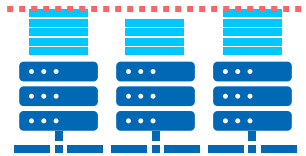
- Store, group, and organize nonzero elements in sparse tensors is a **fundamental problem**
- Goals:
  - reduce tensor **storage**
  - improve **data locality**
  - increase **parallelism**



Motivation

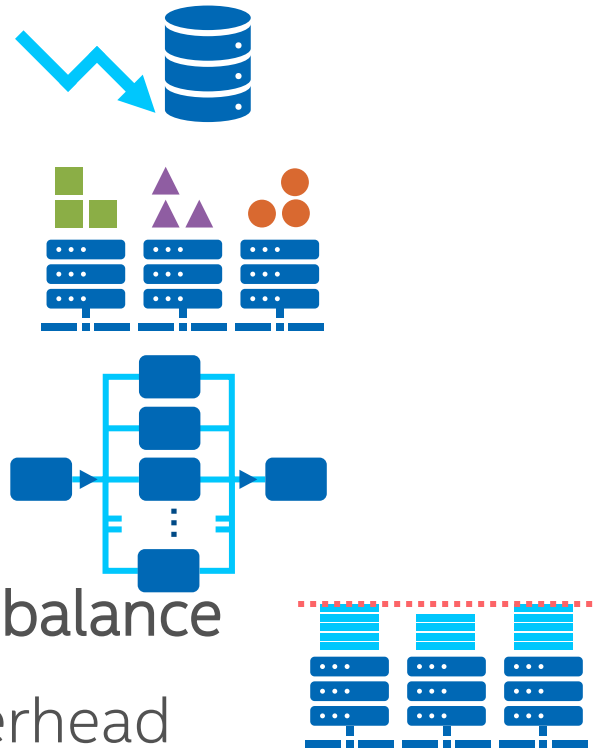
- Store, group, and organize nonzero elements in sparse tensors is a **fundamental problem**
- Goals:
  - reduce tensor **storage**
  - improve **data locality**
  - increase **parallelism**
  - decrease workload **imbalance** & **synchronization** overhead



Motivation

- Store, group, and organize nonzero elements in sparse tensors is a **fundamental problem**
- Goals:
  - ❖ reduce tensor **storage**
  - ❖ improve **data locality**
  - ❖ increase **parallelism**
  - ❖ decrease workload **imbalance** & **synchronization** overhead



Oftentimes conflicting goals

Background

MTTKRP (Matricized Tensor Times Khatri-Rao product)

$$A = X_{(1)}(C \odot B)$$

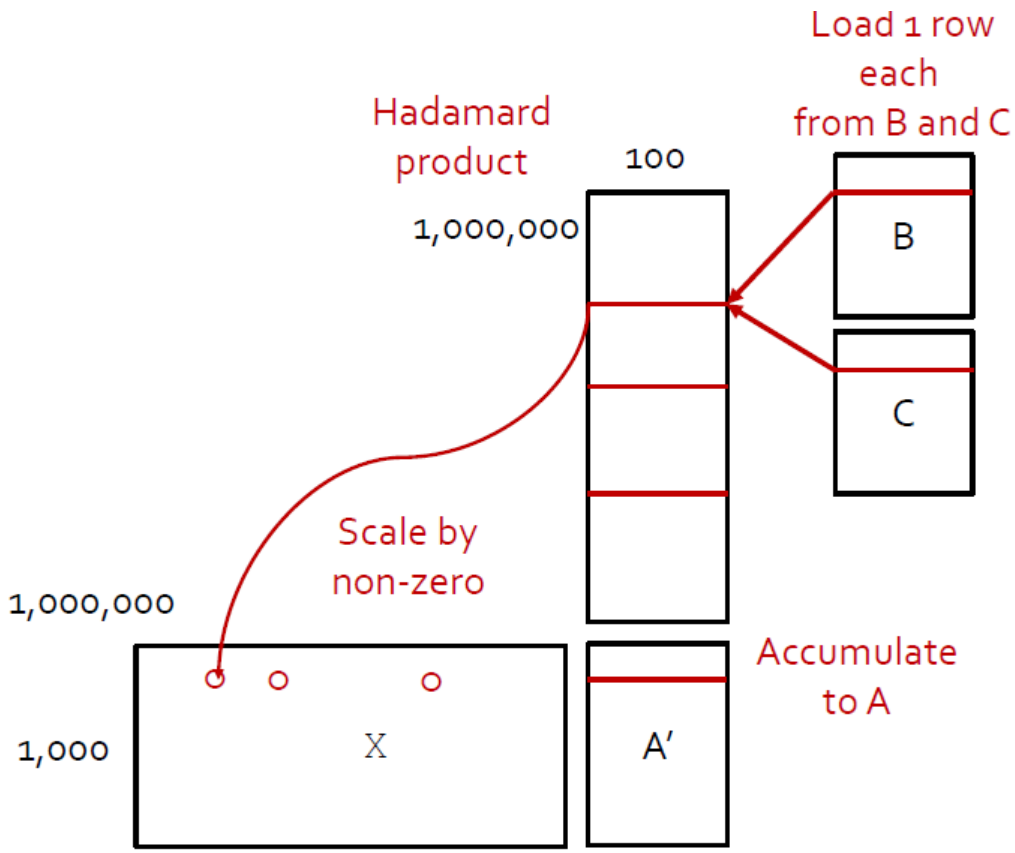
- A key memory-intensive kernel in many sparse tensor computations

Background

MTTKRP (Matricized Tensor Times Khatri-Rao product)

- A key memory-intensive kernel in many sparse tensor computations

$$A = X_{(1)}(C \odot B)$$



Source: Blocking Optimization Strategies for Sparse Tensor Computation, Jee Choi et al.

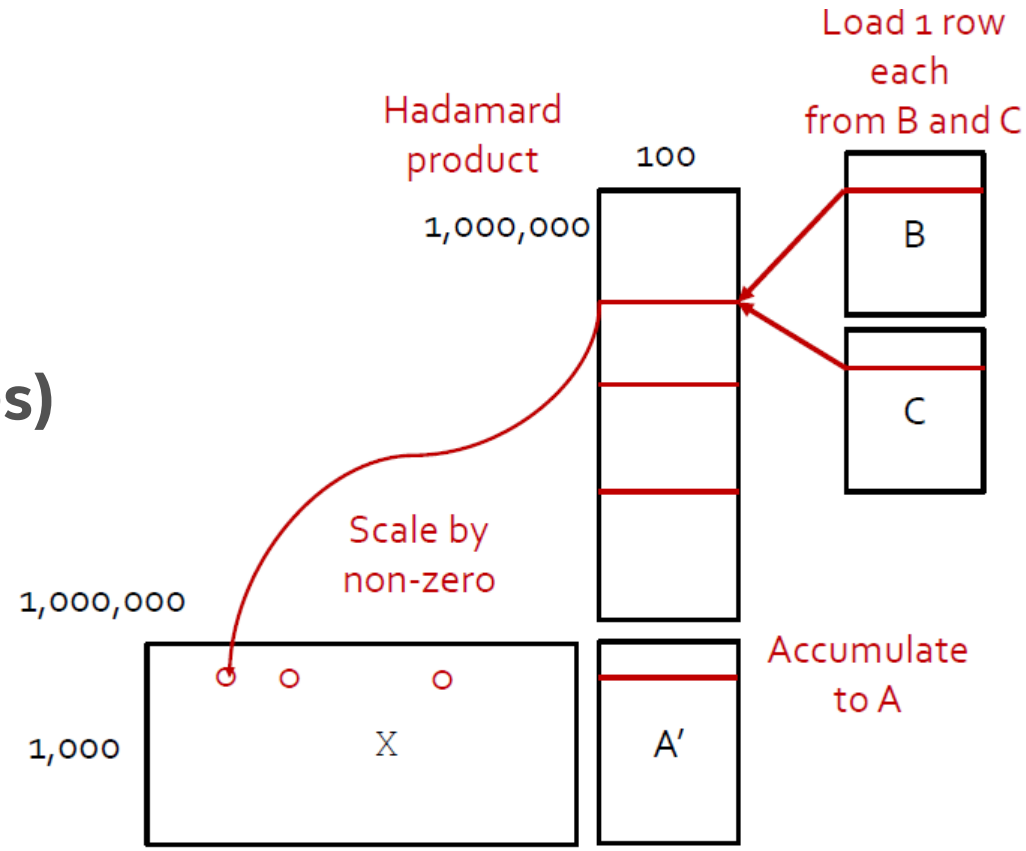
Background

MTTKRP (Matricized Tensor Times Khatri-Rao product)

- A key memory-intensive kernel in many sparse tensor computations

- Challenge: hard to efficiently utilize the memory system for **all dimensions (modes)**

$$A = X_{(1)}(C \odot B)$$



Source: Blocking Optimization Strategies for Sparse Tensor Computation, Jee Choi et al.

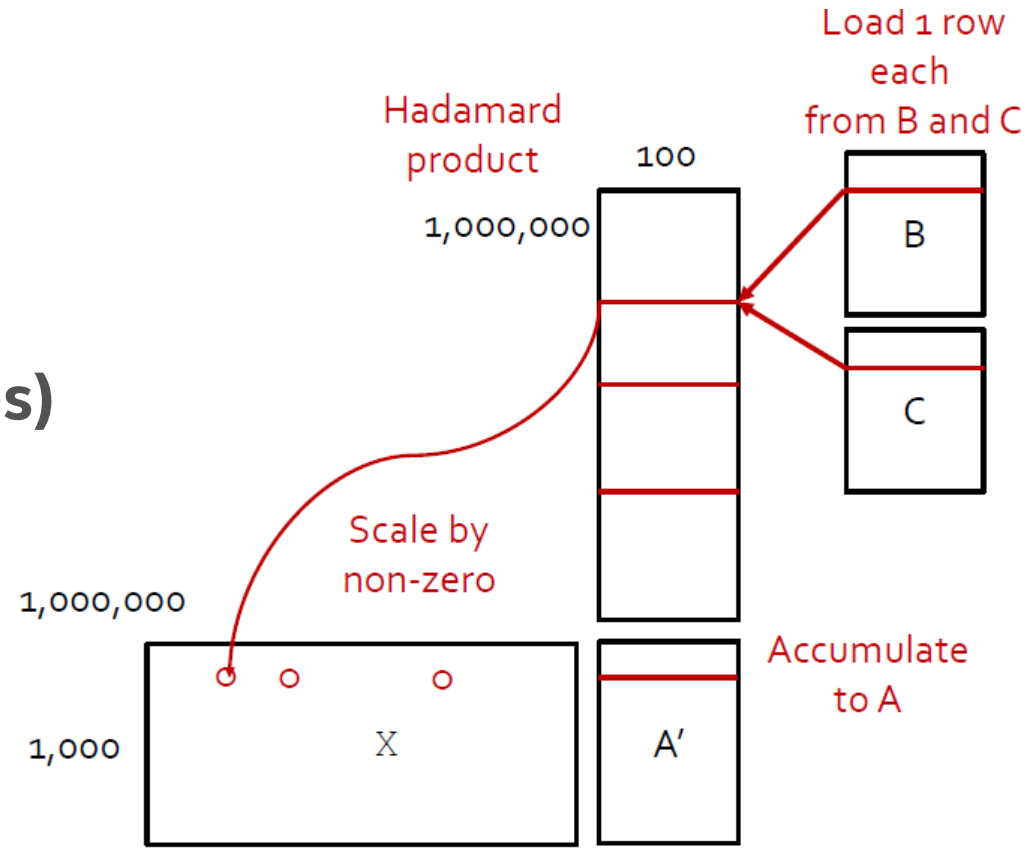
Background

MTTKRP (Matricized Tensor Times Khatri-Rao product)

$$A = X_{(1)}(C \odot B)$$

- A key memory-intensive kernel in many sparse tensor computations

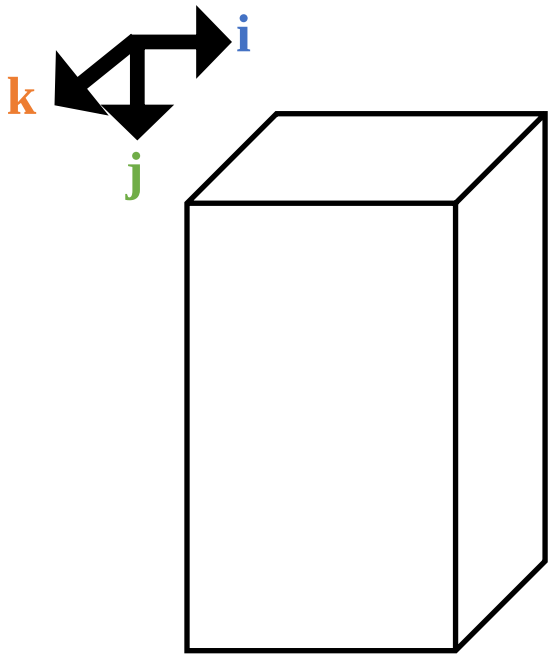
- Challenge: hard to efficiently utilize the memory system for **all dimensions (modes)**
  - **Different data locality patterns across modes**
  - **Update conflicts to factor matrices**
  - **Load imbalance across threads**



Source: Blocking Optimization Strategies for Sparse Tensor Computation, Jee Choi et al.

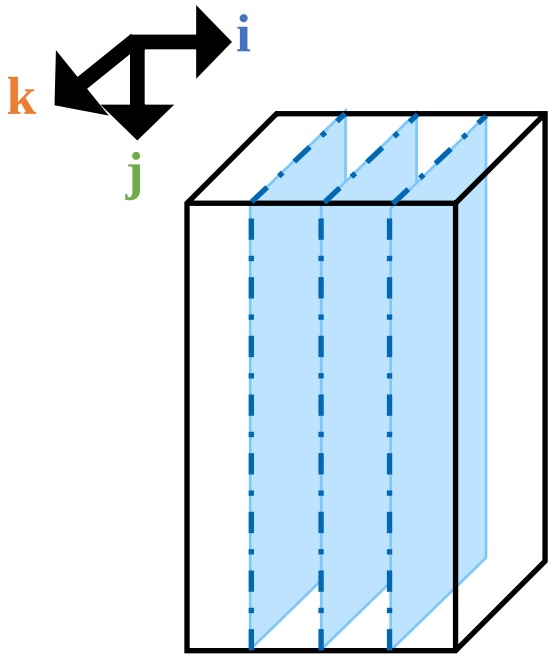
Background

4x8x2 Tensor



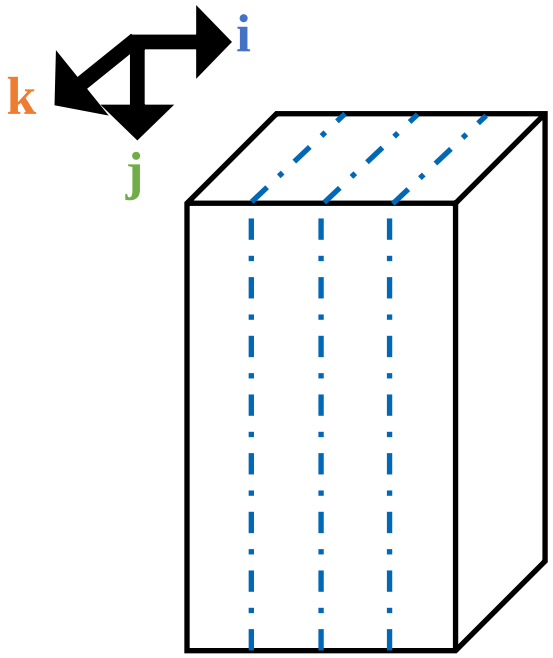
Background

4x8x2 Tensor



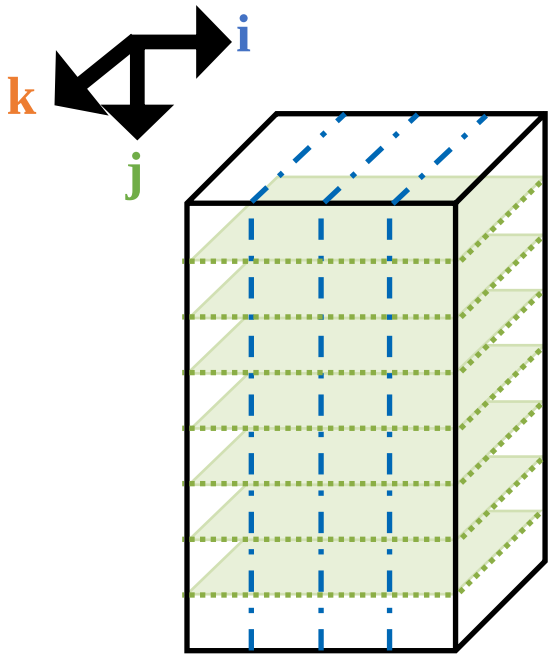
Background

4x8x2 Tensor



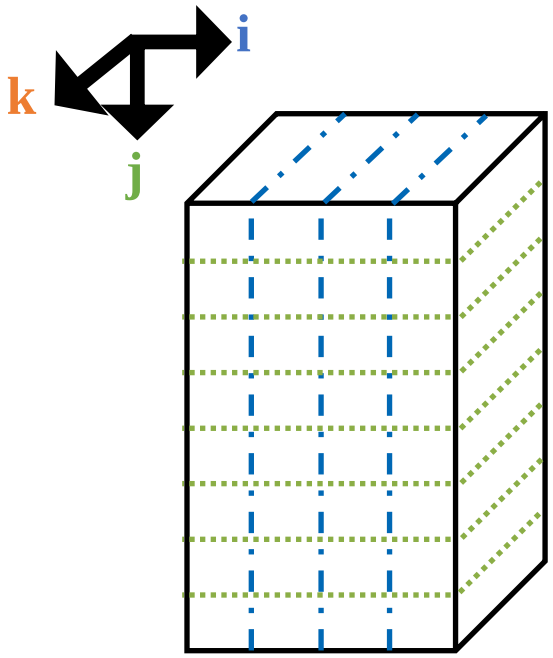
Background

4x8x2 Tensor



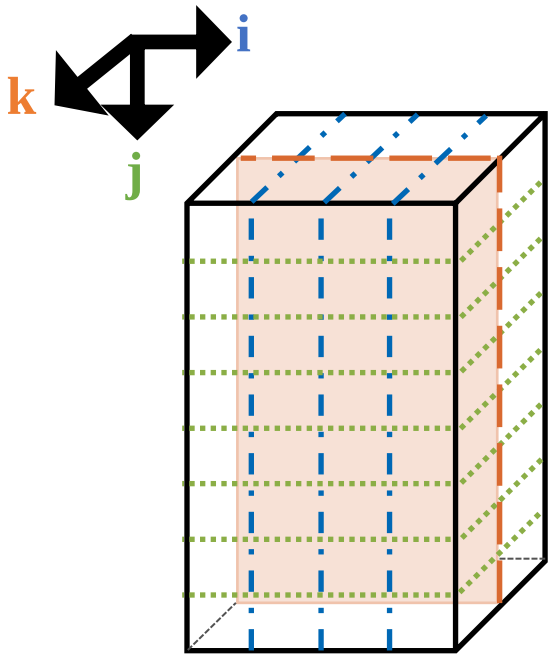
Background

4x8x2 Tensor



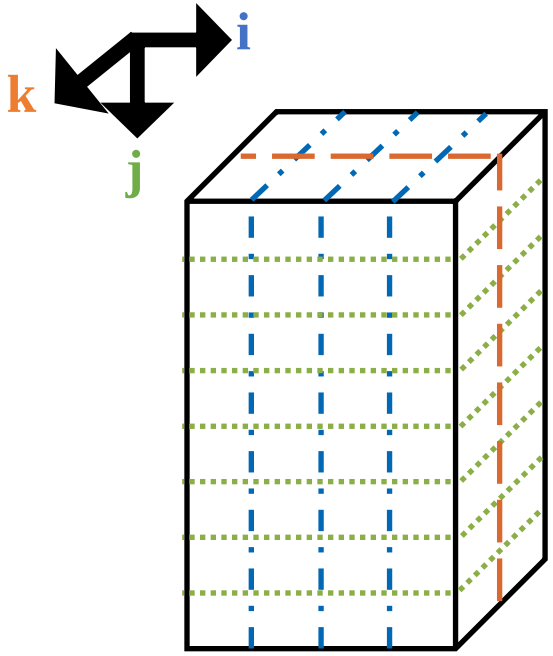
Background

4x8x2 Tensor



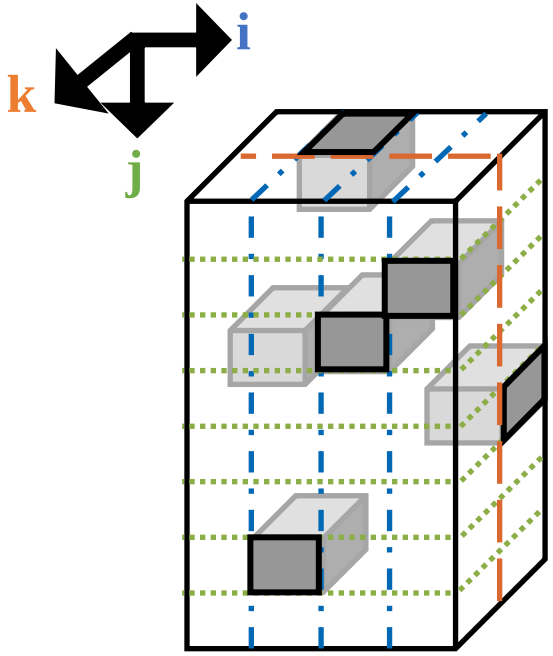
Background

4x8x2 Tensor



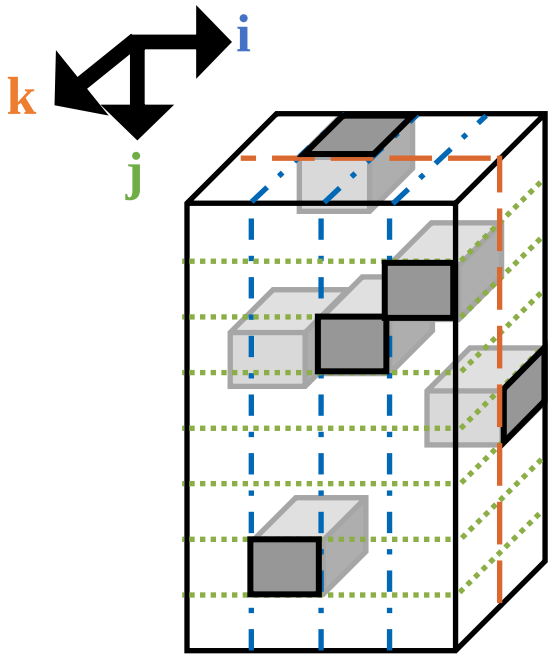
Background

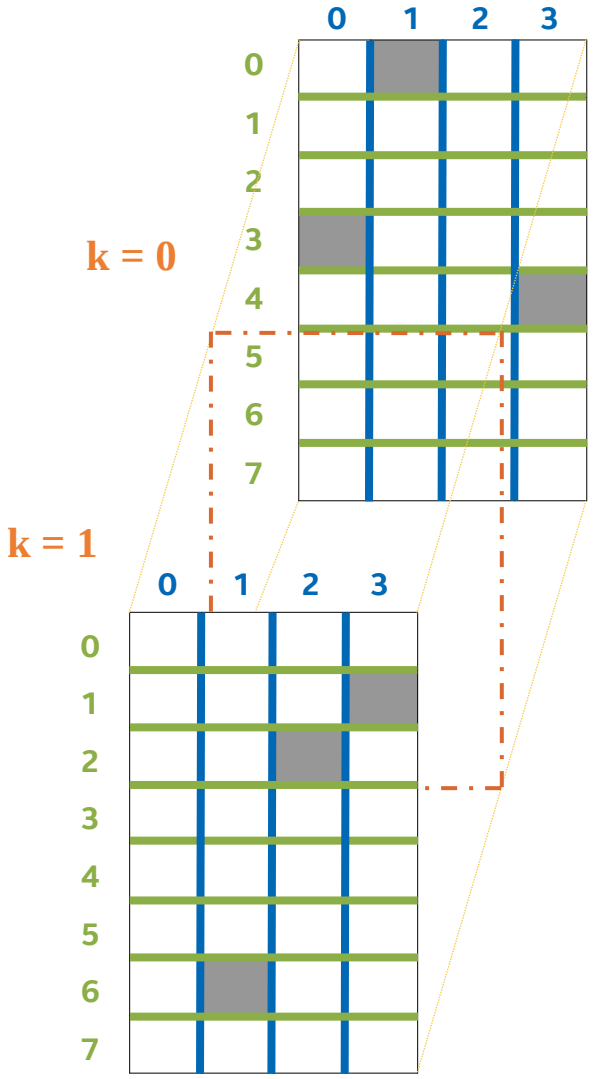
4x8x2 Tensor



Background

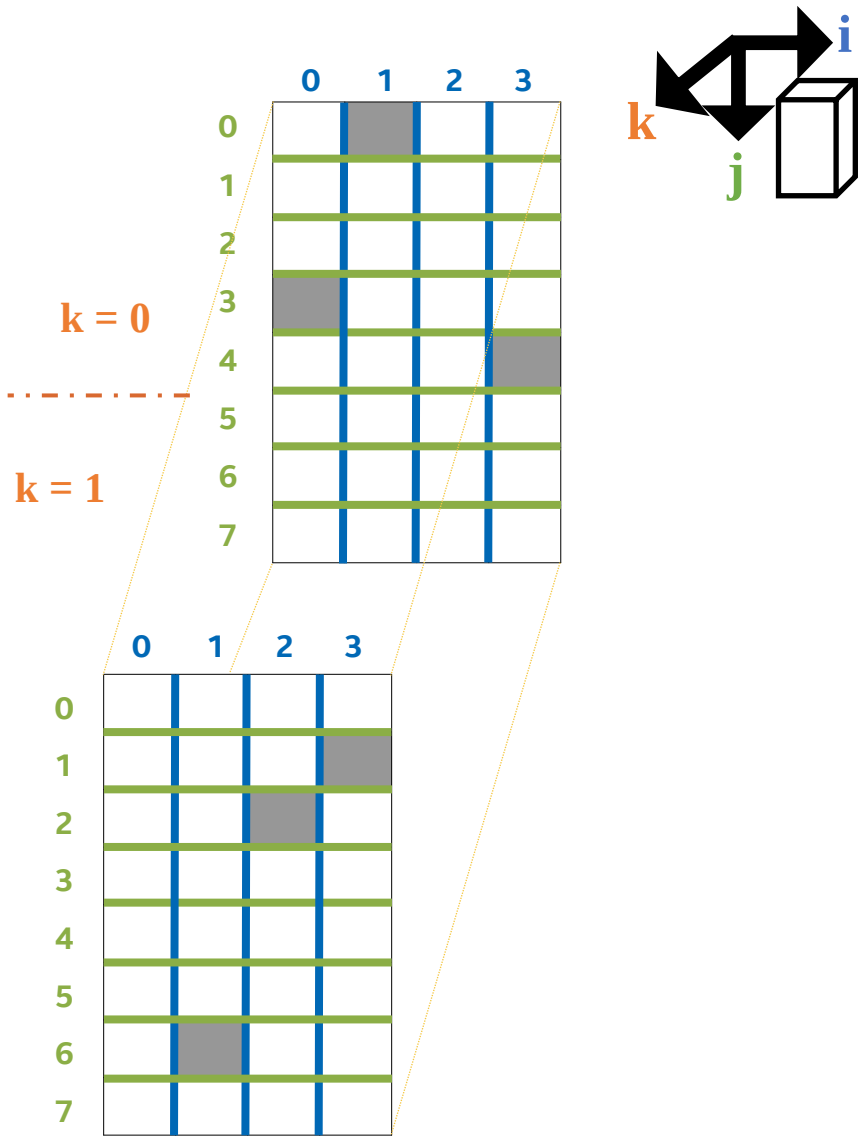
4x8x2 Tensor





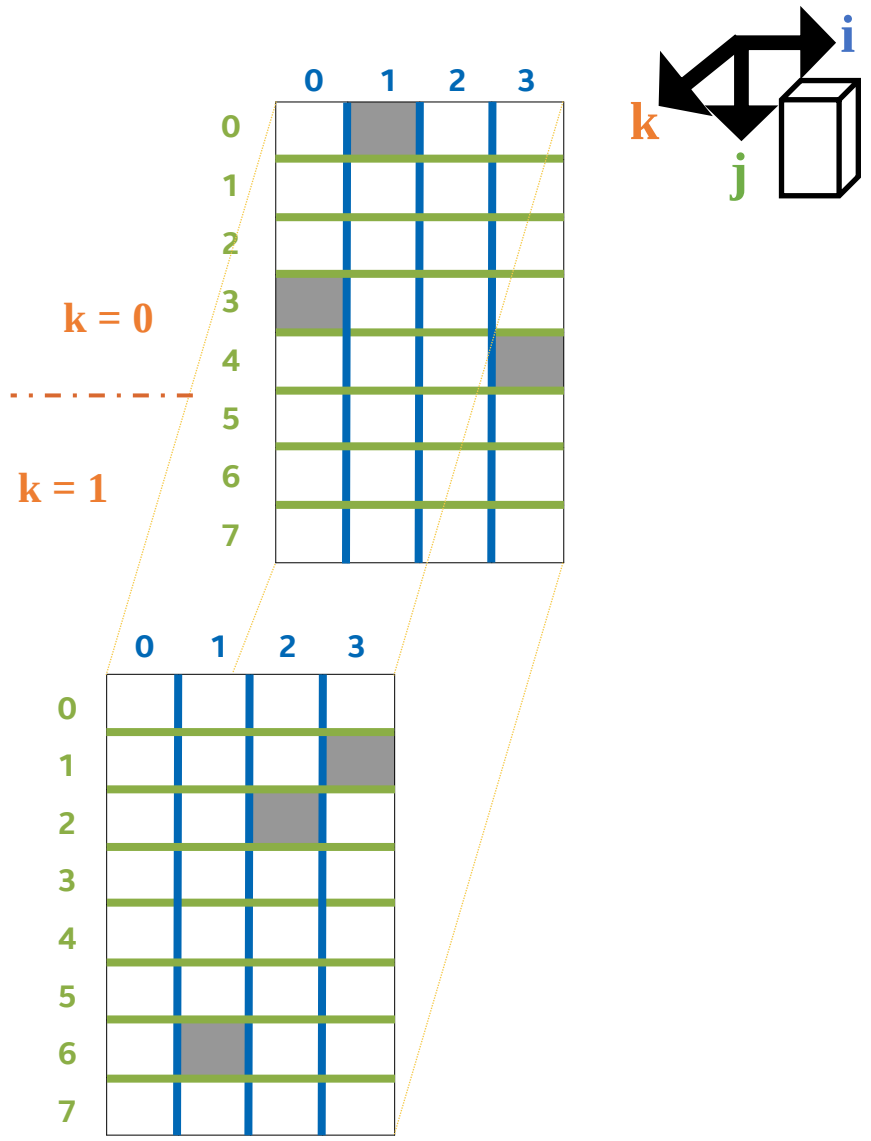
State-of-The-Art Formats

- Can be classified based on their encoding of the indexing metadata into:



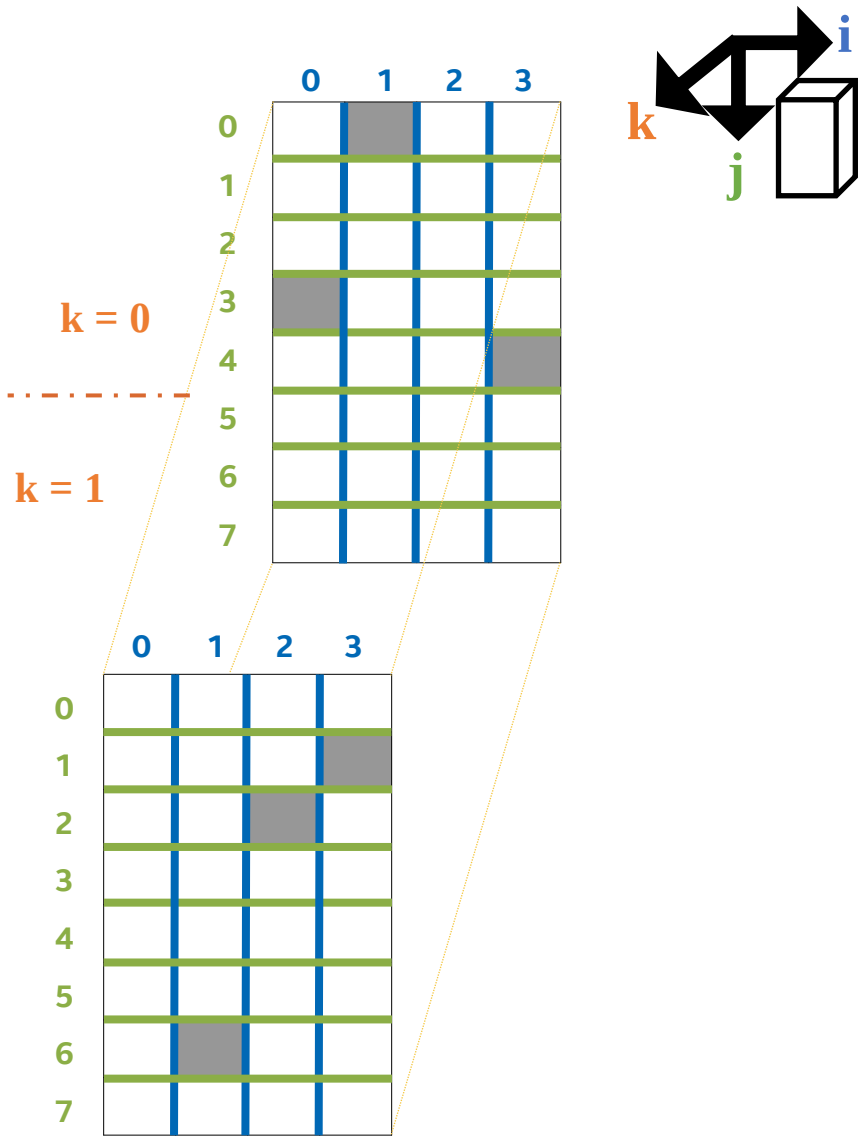
State-of-The-Art Formats

- Can be classified based on their encoding of the indexing metadata into:
  1. List-based formats



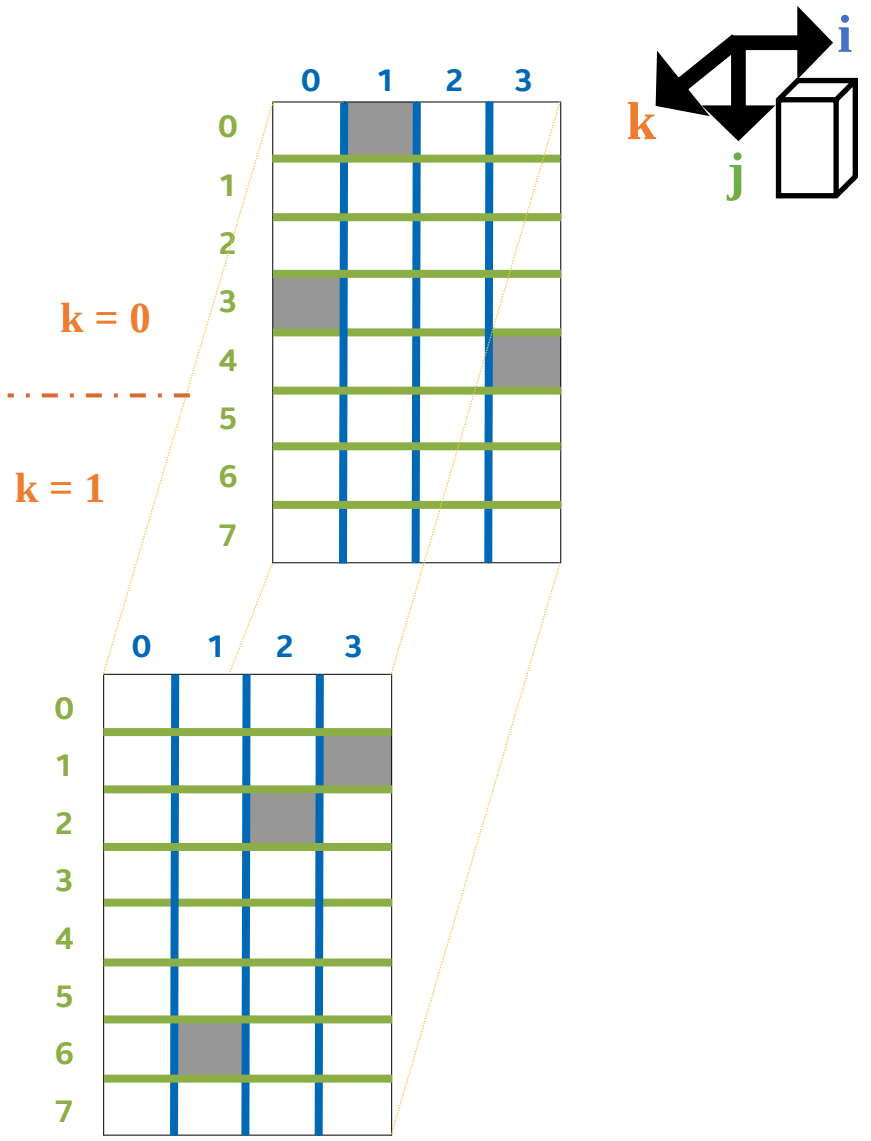
State-of-The-Art Formats

- Can be classified based on their encoding of the indexing metadata into:
  1. List-based formats
  2. Tree-based formats



State-of-The-Art Formats

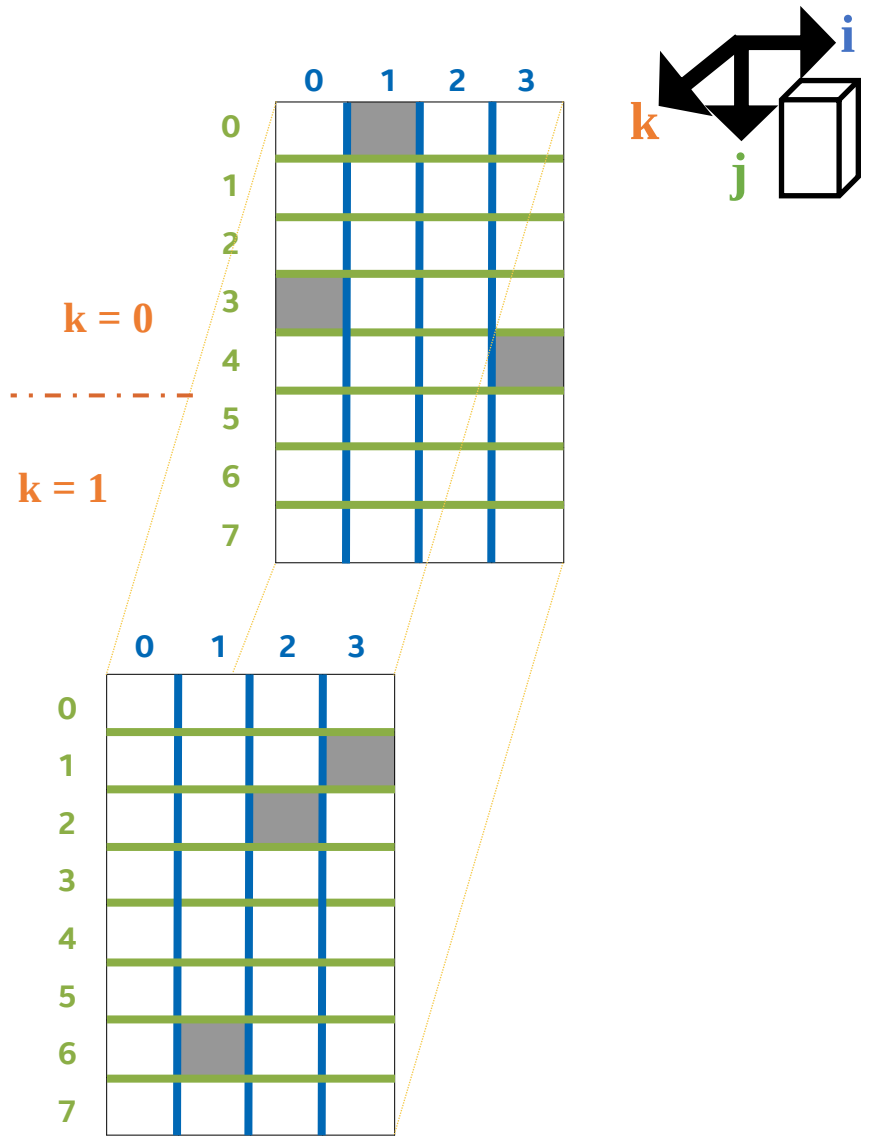
- Can be classified based on their encoding of the indexing metadata into:
  1. List-based formats
  2. Tree-based formats
  3. Block-based formats



State-of-The-Art Formats

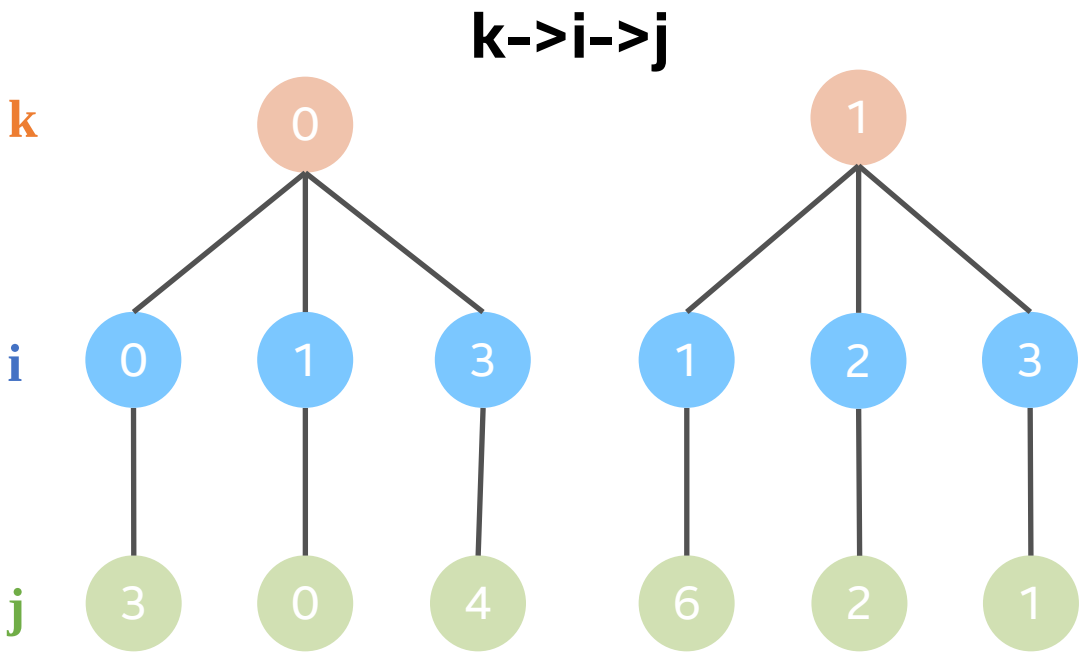
- COO (coordinate): list-based format

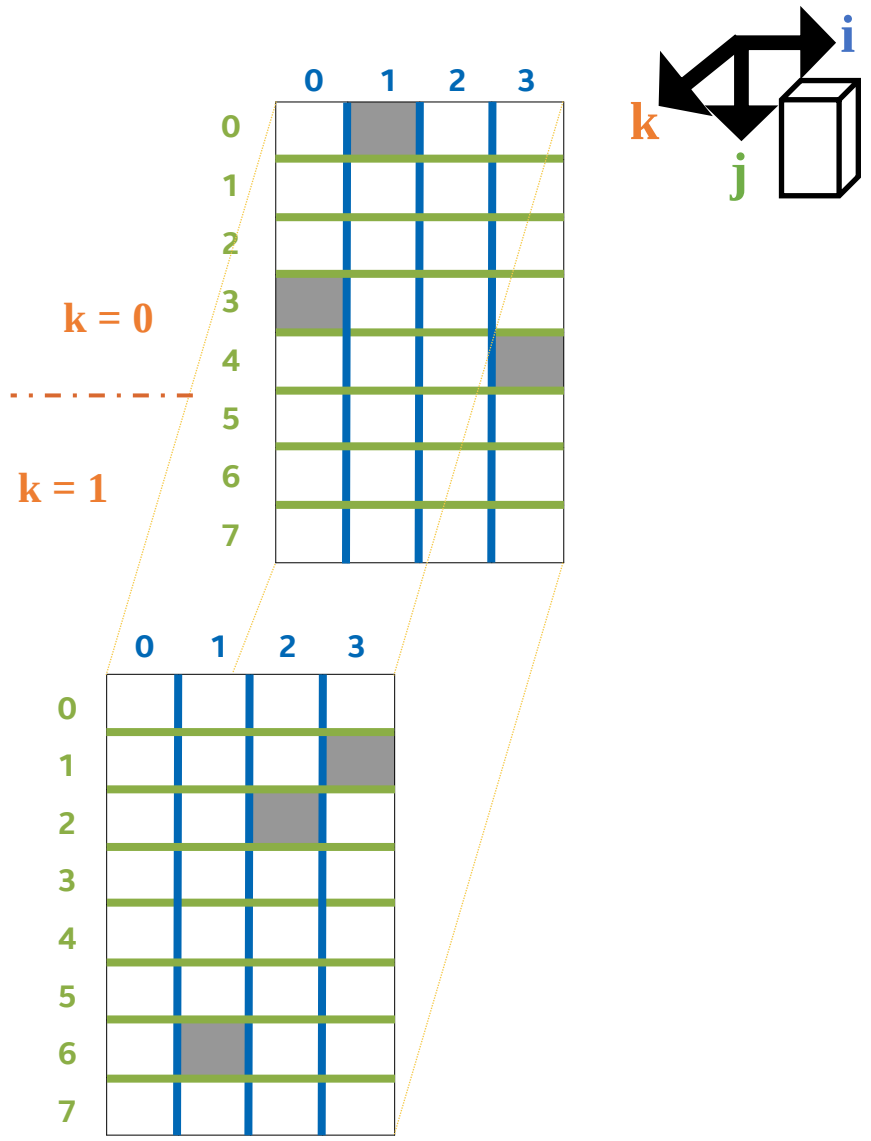
i	j	k
0	3	0
1	0	0
1	6	1
2	2	1
3	1	1
3	4	0



State-of-The-Art Formats

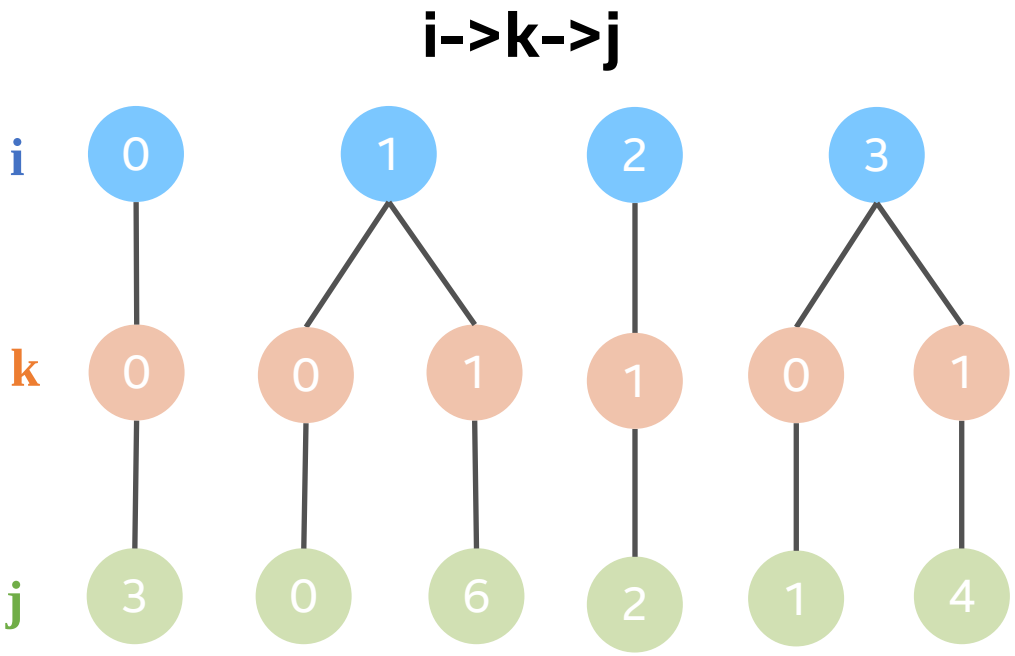
- CSF (compressed sparse fiber): **tree-based format**

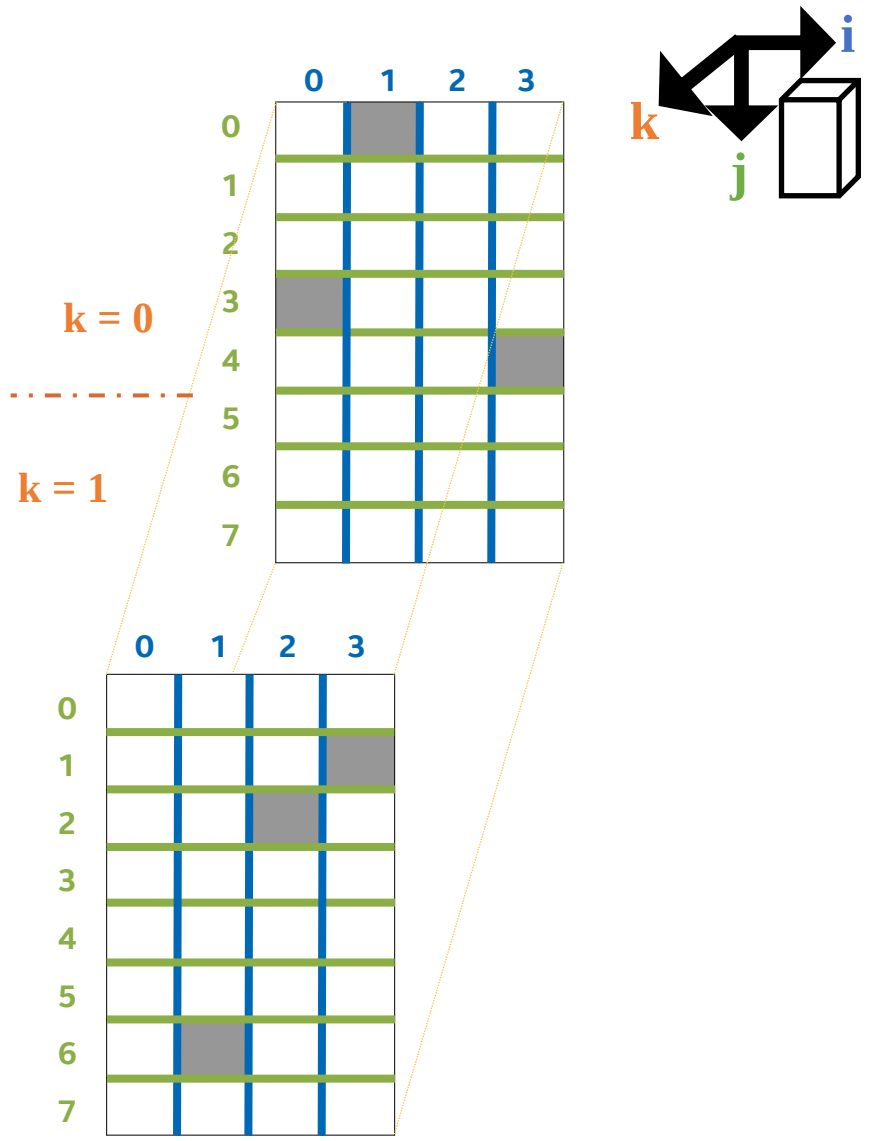




State-of-The-Art Formats

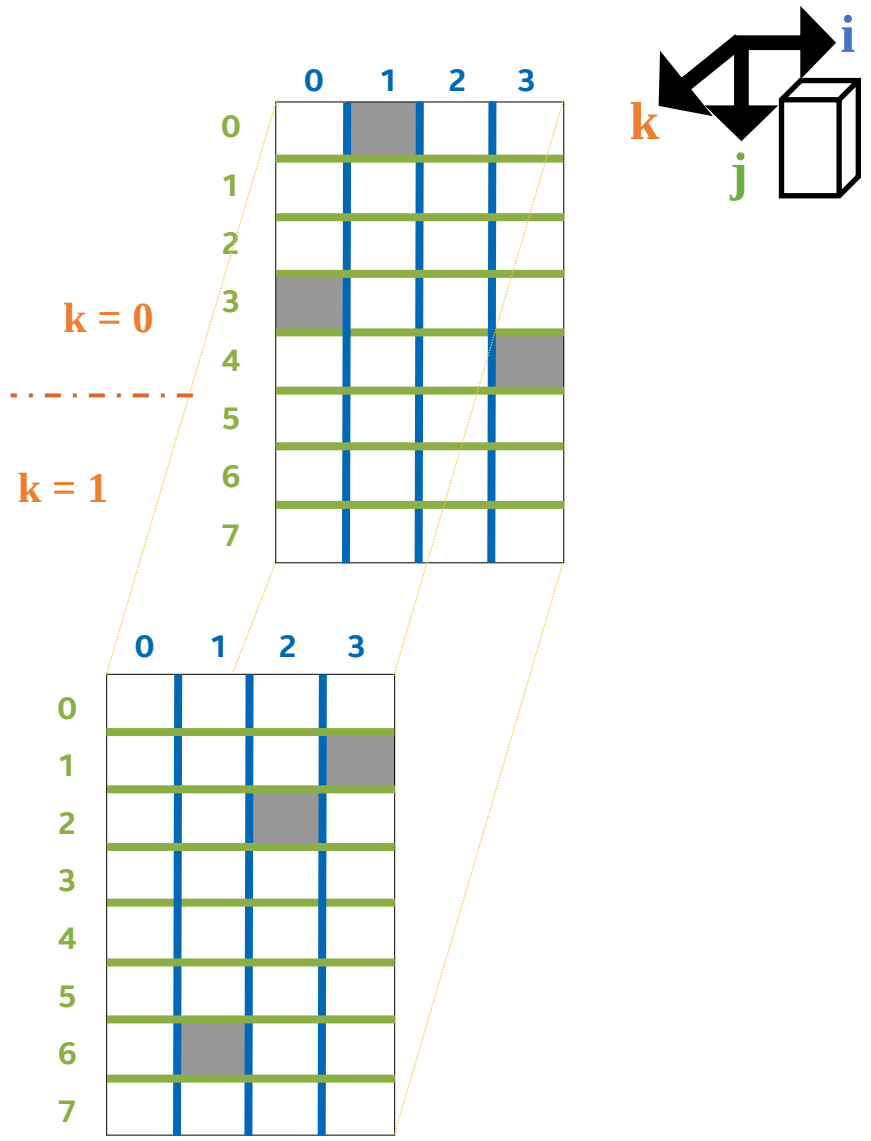
- CSF (compressed sparse fiber): **tree-based format**





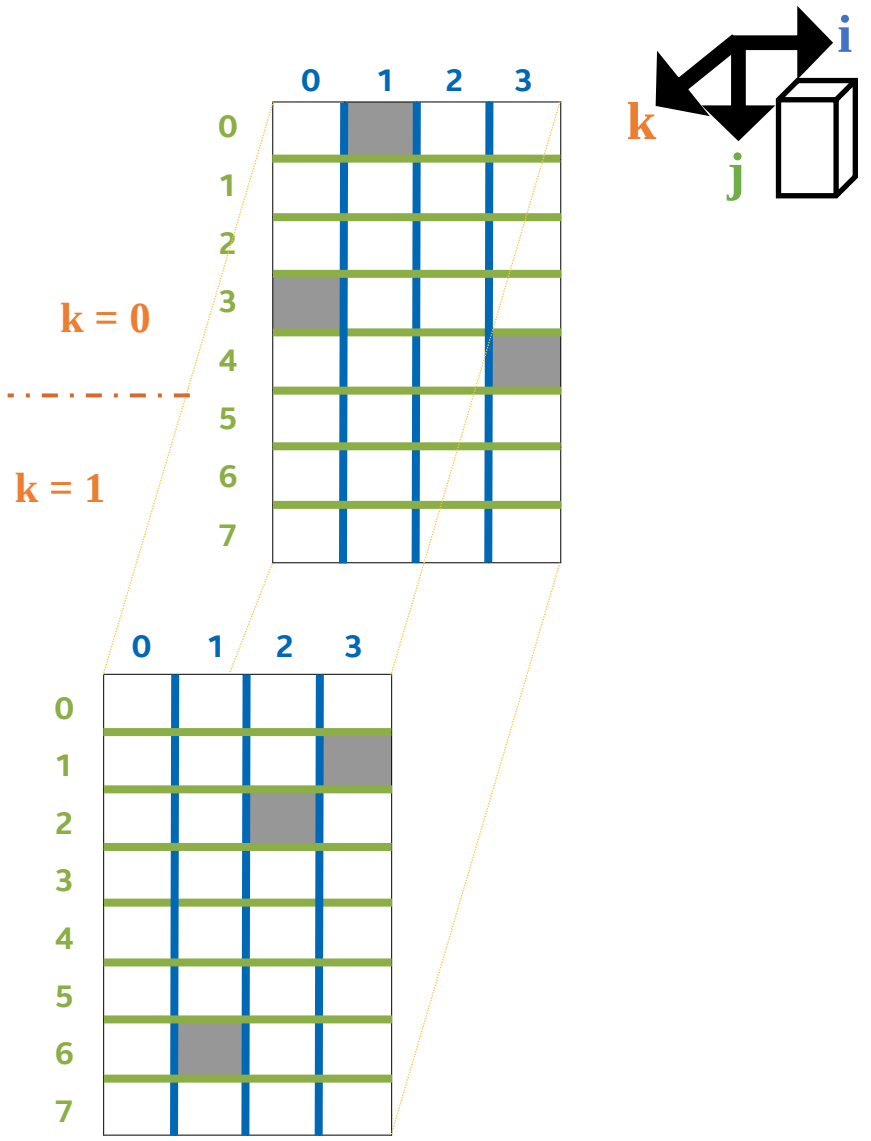
State-of-The-Art Formats

- CSF (compressed sparse fiber): **tree-based format**
  - 6 possible representations for a 3D tensor



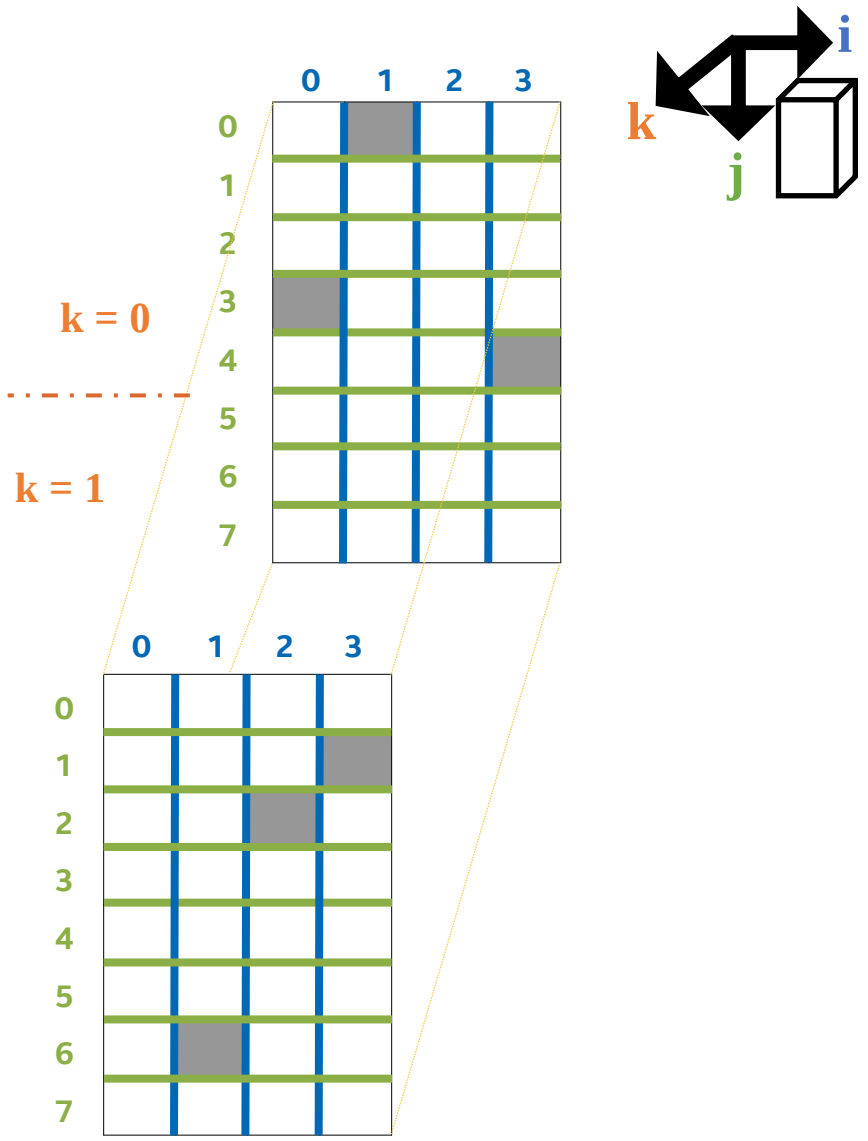
State-of-The-Art Formats

- CSF (compressed sparse fiber): **tree-based format**
  - 6 possible representations for a 3D tensor
  - In practice, we only keep 3 representations



State-of-The-Art Formats

- CSF (compressed sparse fiber): **tree-based format**
  - 6 possible representations for a 3D tensor
  - In practice, we only keep 3 representations
    1. i -> k -> j
    2. k -> i -> j
    3. j -> k -> i

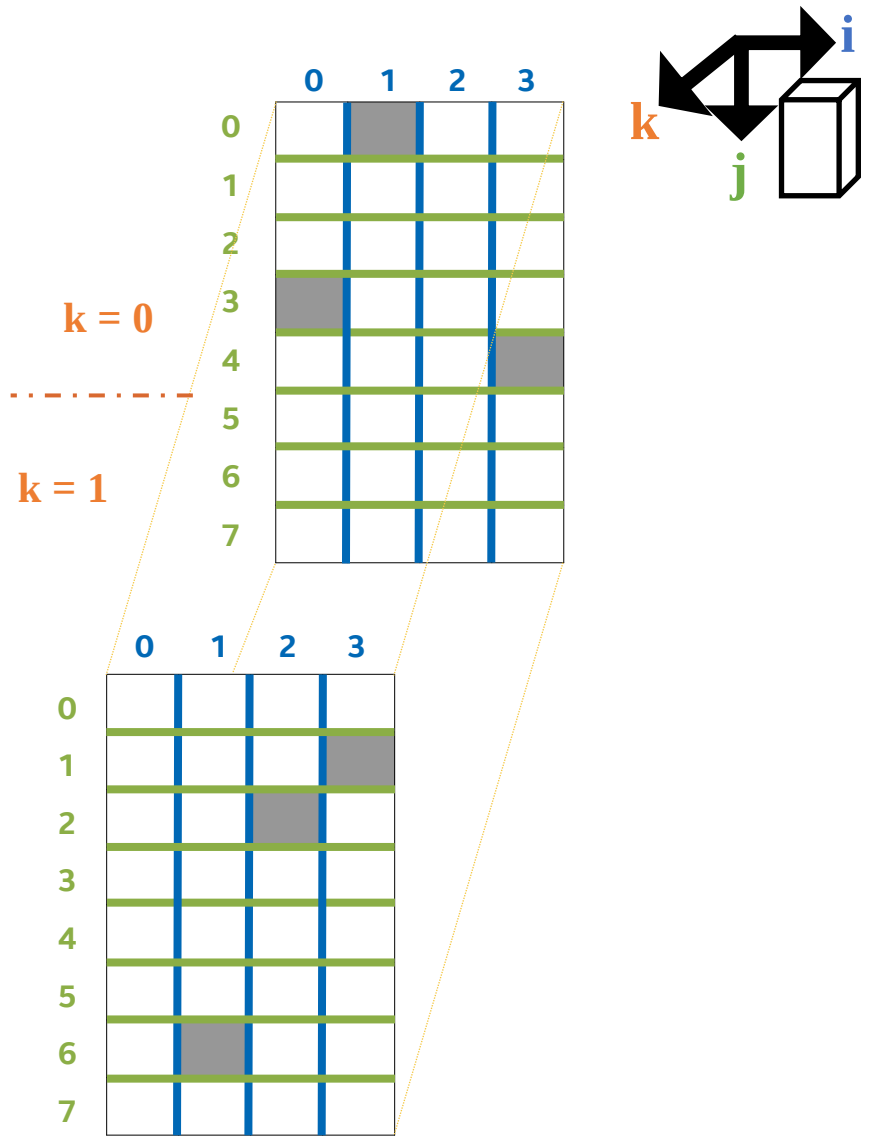


State-of-The-Art Formats

- HiCOO (hierarchical coordinate): **block-based format**
  - size(e-idx) << size(b-idx)

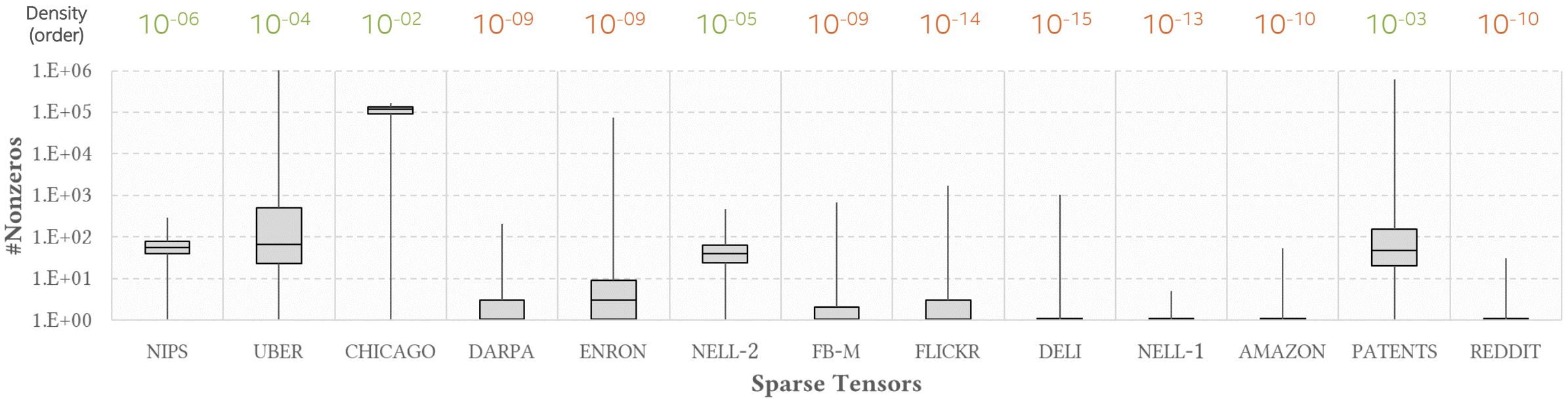
2X2X2

bptr	b_i	b_j	b_k	e_i	e_j	e_k
0	0	0	0	1	0	0
1	0	1	0	0	1	0
2	0	3	0	1	0	1
3	1	0	0	1	1	1
4	1	1	0	0	0	1
5	1	2	0	1	0	0



State-of-The-Art Formats

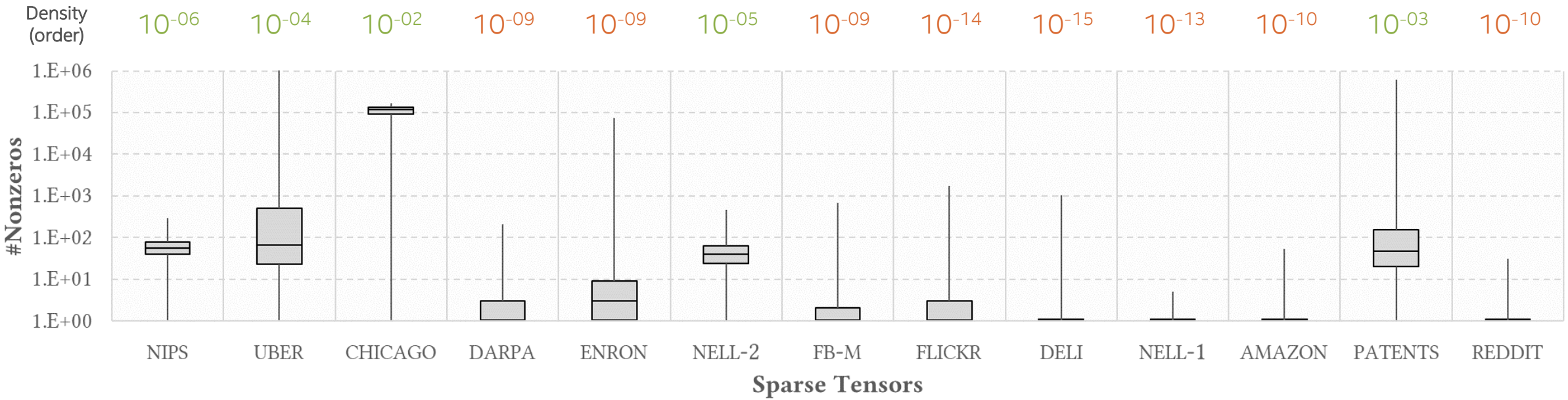
Spatial data distribution



A box plot of the data (nonzero elements) distribution across multi-dimensional blocks (subspaces). The multi-dimensional subspace size is 128^N, where N is the number of dimensions (modes).

State-of-The-Art Formats

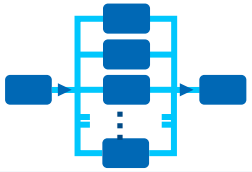
Spatial data distribution



A box plot of the data (nonzero elements) distribution across multi-dimensional blocks (subspaces). The multi-dimensional subspace size is 128^N, where N is the number of dimensions (modes).

increased sparsity → inefficient memory usage

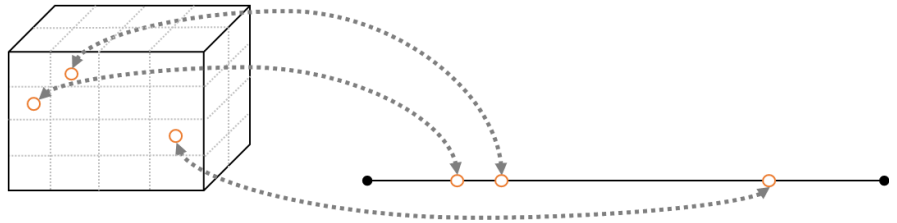
State-of-The-Art Formats


Formats	Granularity	Mode Orientation	Data Locality	Parallelism	Load balance
List-based (COO)	NNZ element	Mode-agnostic	NA	Suffer from conflicts	Maximized
Tree-based (CSF)	Compressed sparse fiber	Mode-specific	Improved for specific mode	Improved for specific mode	Work imbalance (especially in short modes)
Block-based (HiCOO)	Compressed block	Mode-agnostic	Improved	Suffer from conflicts	Work imbalance across blocks

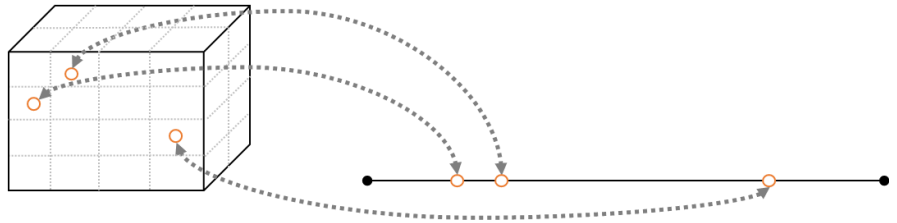
Adaptive Linearized Tensor Order (**ALTO**)

- A new storage format for high performance and scalable sparse tensor computations
- Maps a multi-dimensional space into a linear space, and vice versa, using an adaptive (**data-aware**) encoding scheme



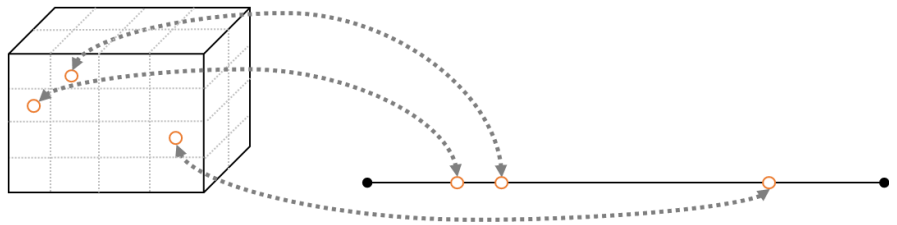
Adaptive Linearized Tensor Order (**ALTO**)

- A new storage format for high performance and scalable sparse tensor computations
- Maps a multi-dimensional space into a linear space, and vice versa, using an adaptive (**data-aware**) encoding scheme
  - ➢ mode-agnostic linearized index for each NNZ



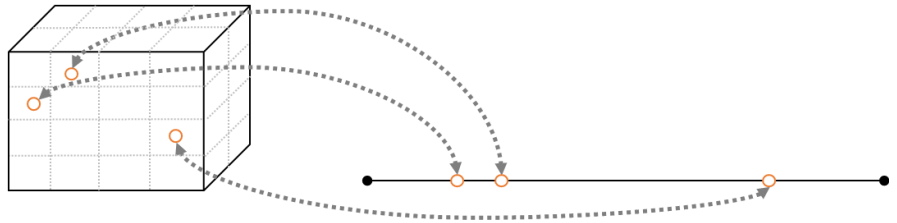
Adaptive Linearized Tensor Order (**ALTO**)

- A new storage format for high performance and scalable sparse tensor computations
- Maps a multi-dimensional space into a linear space, and vice versa, using an adaptive (**data-aware**) encoding scheme
  - ➢ mode-agnostic linearized index for each NNZ
  - ➢ compact representation using the minimum number of bits



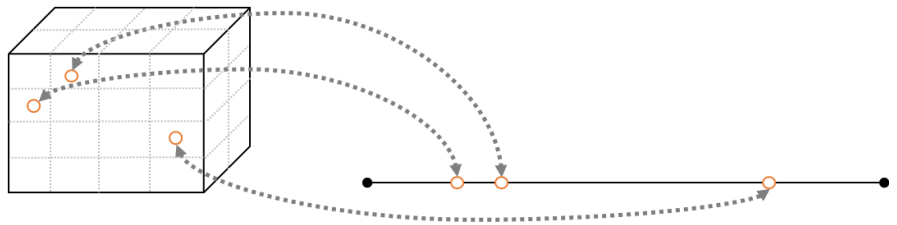
Adaptive Linearized Tensor Order (**ALTO**)

- A new storage format for high performance and scalable sparse tensor computations
- Maps a multi-dimensional space into a linear space, and vice versa, using an adaptive (**data-aware**) encoding scheme
  - ➢ mode-agnostic linearized index for each NNZ
  - ➢ compact representation using the minimum number of bits
  - ➢ improved data locality across all modes



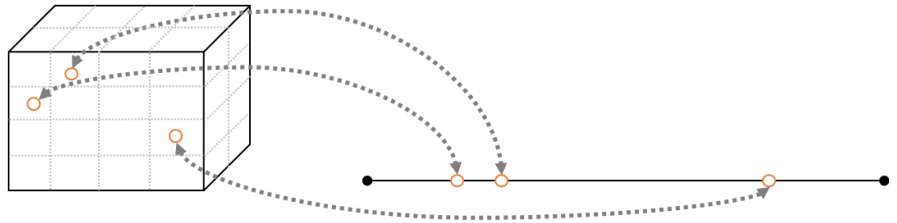
Adaptive Linearized Tensor Order (**ALTO**)

- A new storage format for high performance and scalable sparse tensor computations
- Maps a multi-dimensional space into a linear space, and vice versa, using an adaptive (**data-aware**) encoding scheme
  - ➢ mode-agnostic linearized index for each NNZ
  - ➢ compact representation using the minimum number of bits
  - ➢ improved data locality across all modes
  - ➢ perfectly balanced partitions by NNZs for effective scaling

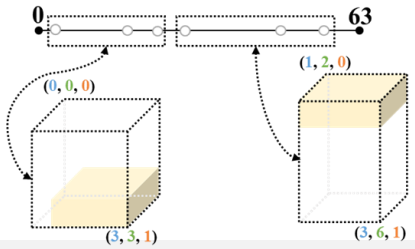


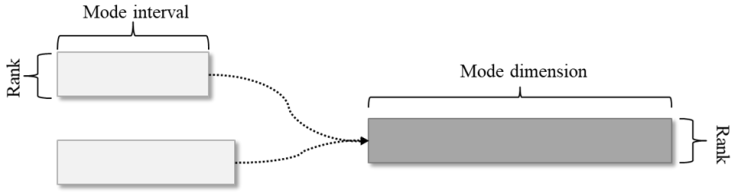
Adaptive Linearized Tensor Order (**ALTO**)

- A new storage format for high performance and scalable sparse tensor computations
- Maps a multi-dimensional space into a linear space, and vice versa, using an adaptive (**data-aware**) encoding scheme
  - ➢ mode-agnostic linearized index for each NNZ
  - ➢ compact representation using the minimum number of bits
  - ➢ improved data locality across all modes
  - ➢ perfectly balanced partitions by NNZs for effective scaling

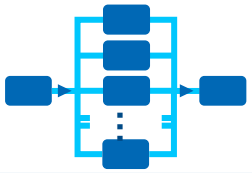


- **Data-aware** synchronization to resolve conflicting updates (writes) across threads





Adaptive Linearized Tensor Order (ALTO)

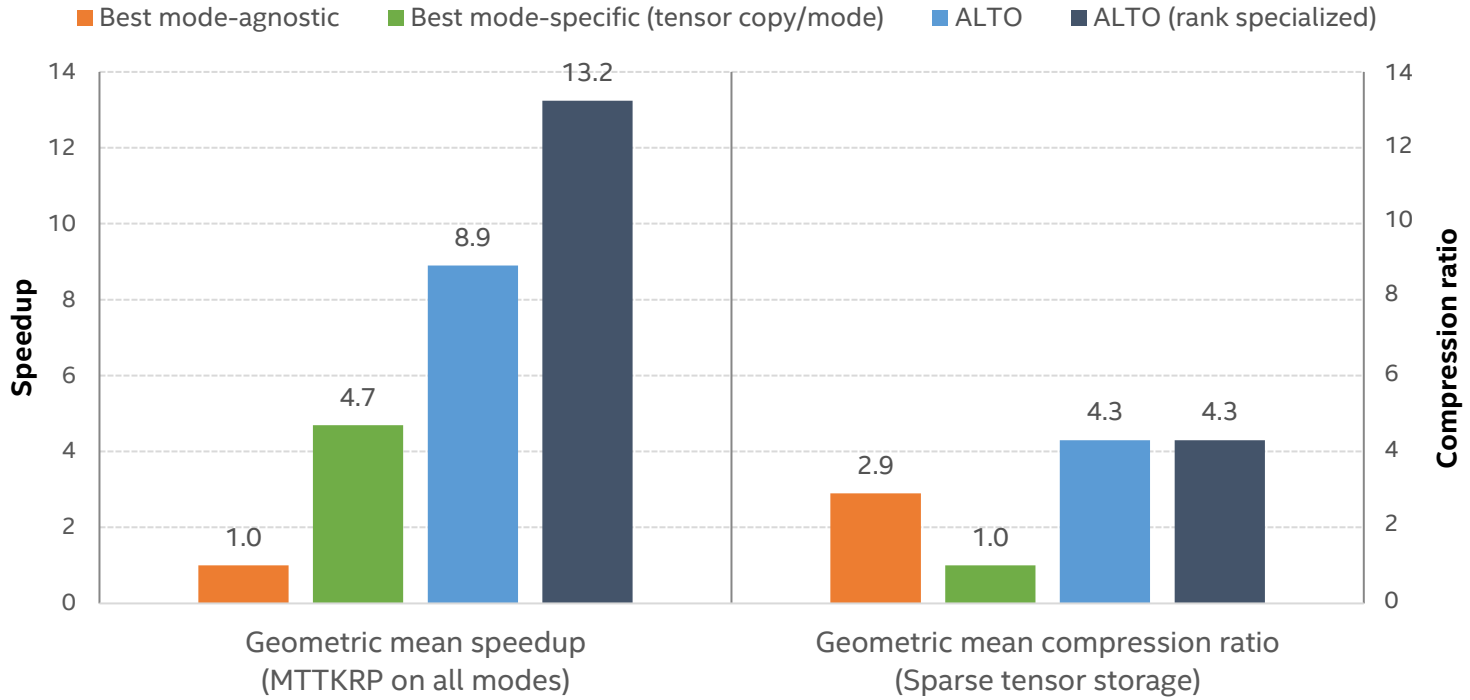




Formats	Granularity	Mode Orientation	Data Locality	Parallelism	Load balance
List-based (COO)	NNZ element	Mode-agnostic	NA	Suffer from conflicts	Maximized
Tree-based (CSF)	Compressed sparse fiber	Mode-specific	Improved for specific mode	Improved for specific mode	Work imbalance (especially in short modes)
Block-based (HiCOO)	Compressed block	Mode-agnostic	Improved	Suffer from conflicts	Work imbalance across blocks
ALTO	NNZ element	Mode-agnostic	Improved	Improved	Maximized

Adaptive Linearized Tensor Order (ALTO)

- ALTO **outperforms** an oracle that selects the best state-of-the-art format in terms of the parallel performance and tensor storage

- **Platform**: an Intel Xeon Platinum 8280 (CLX) w/ 2x28 cores
- **Mode-agnostic formats**: coordinate (COO) and hierarchical COO (HiCOO)
- **Mode-specific formats**: compressed sparse fiber (CSF) and tiled CSF

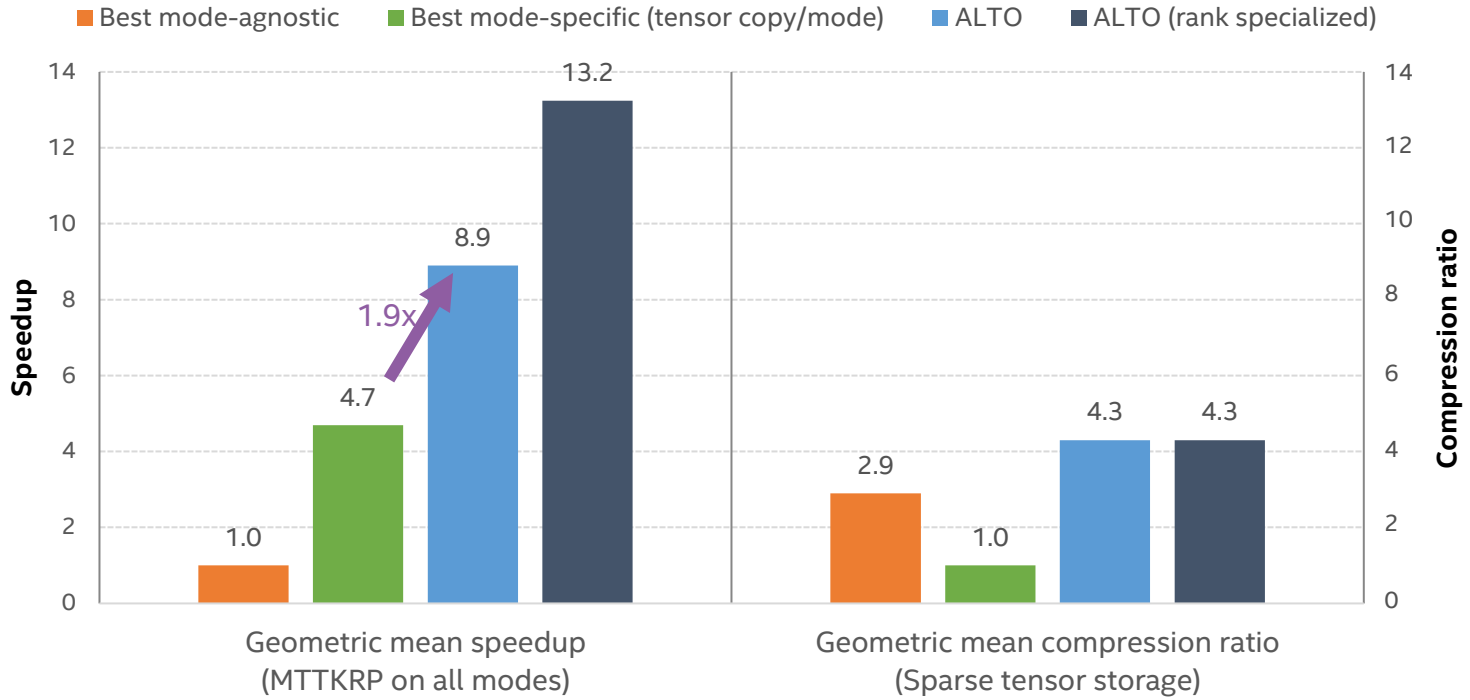


"ALTO: Adaptive Linearized Storage of Sparse Tensors", Ahmed Helal, Jan Laukemann, Fabio Checconi, Jesmin Jahan Tithi, Teresa Ranadive, Fabrizio Petrini, Jee Choi, **accepted** to ACM International Conference on Supercomputing (ICS), 2021

Adaptive Linearized Tensor Order (ALTO)

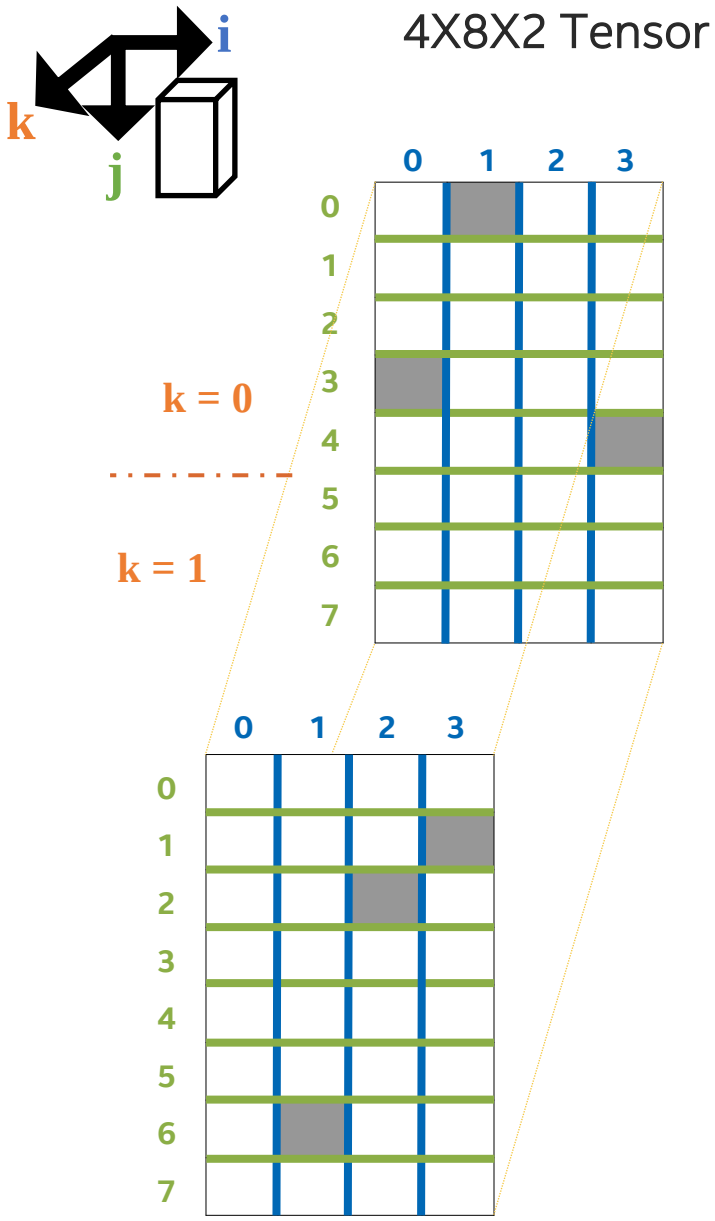
- ALTO **outperforms** an oracle that selects the best state-of-the-art format in terms of the parallel performance and tensor storage

- **Platform**: an Intel Xeon Platinum 8280 (CLX) w/ 2x28 cores
- **Mode-agnostic formats**: coordinate (COO) and hierarchical COO (HiCOO)
- **Mode-specific formats**: compressed sparse fiber (CSF) and tiled CSF

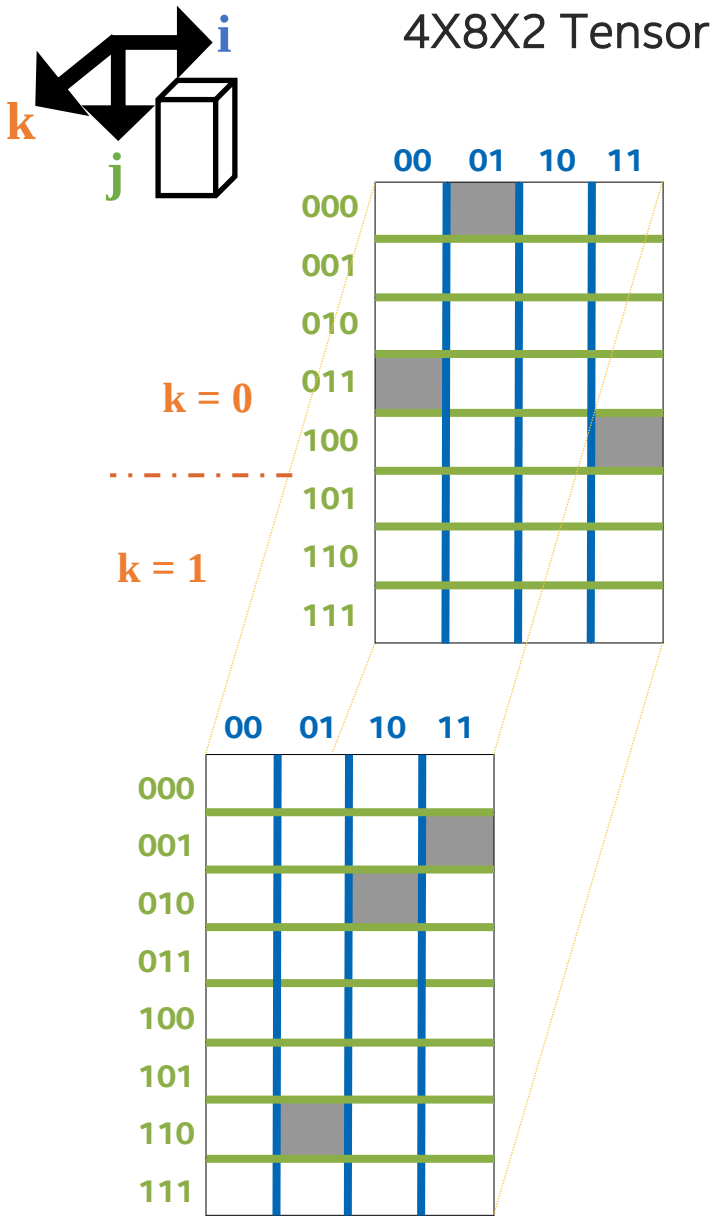


"ALTO: Adaptive Linearized Storage of Sparse Tensors", Ahmed Helal, Jan Laukemann, Fabio Checconi, Jesmin Jahan Tithi, Teresa Ranadive, Fabrizio Petrini, Jee Choi, **accepted** to ACM International Conference on Supercomputing (ICS), 2021

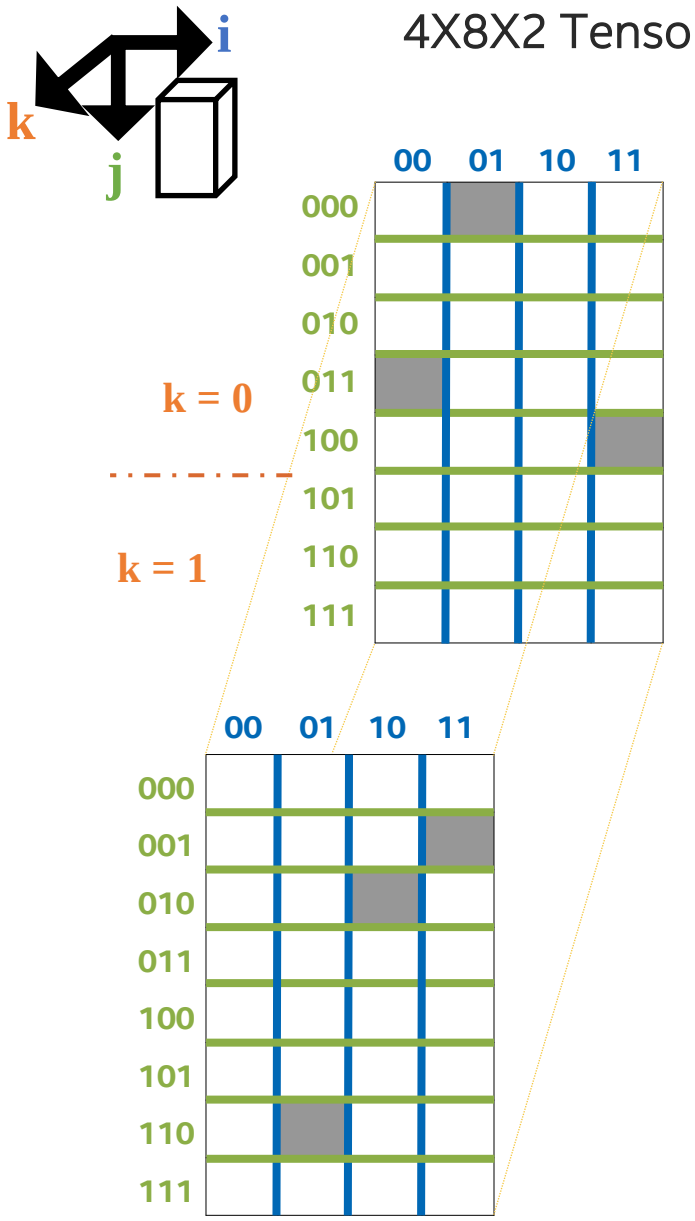
ALTO: Example



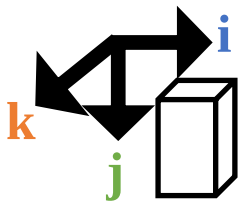
ALTO: Example



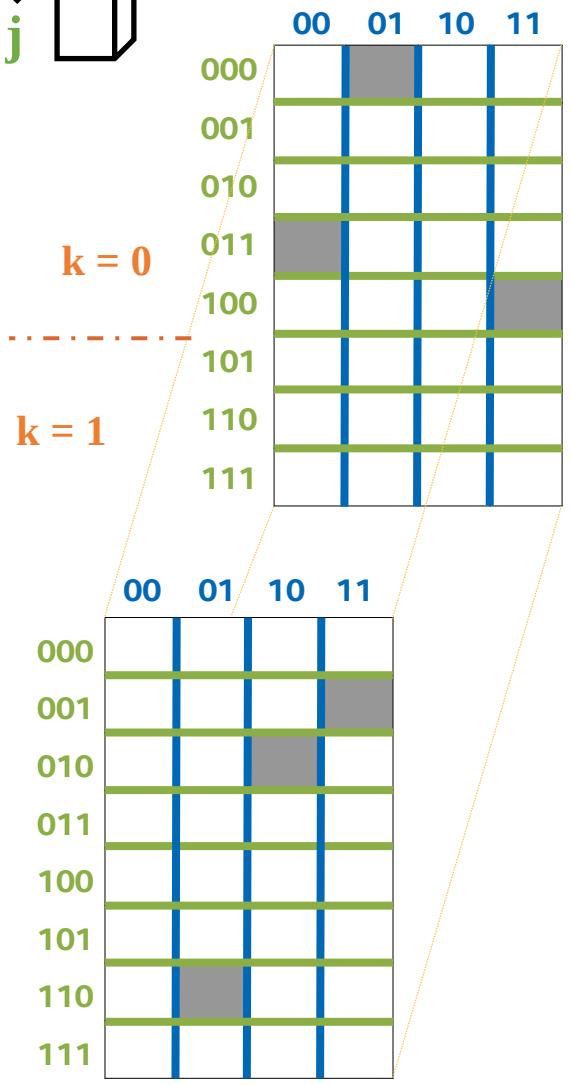
ALTO: Example



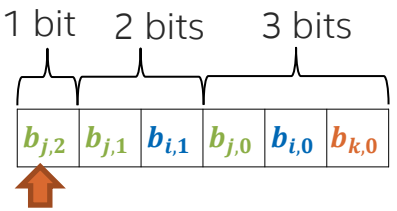
ALTO: Example

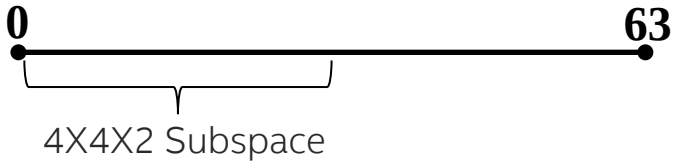


4X8X2 Tensor

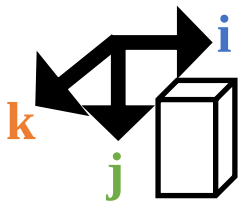


ALTO Bit Mask

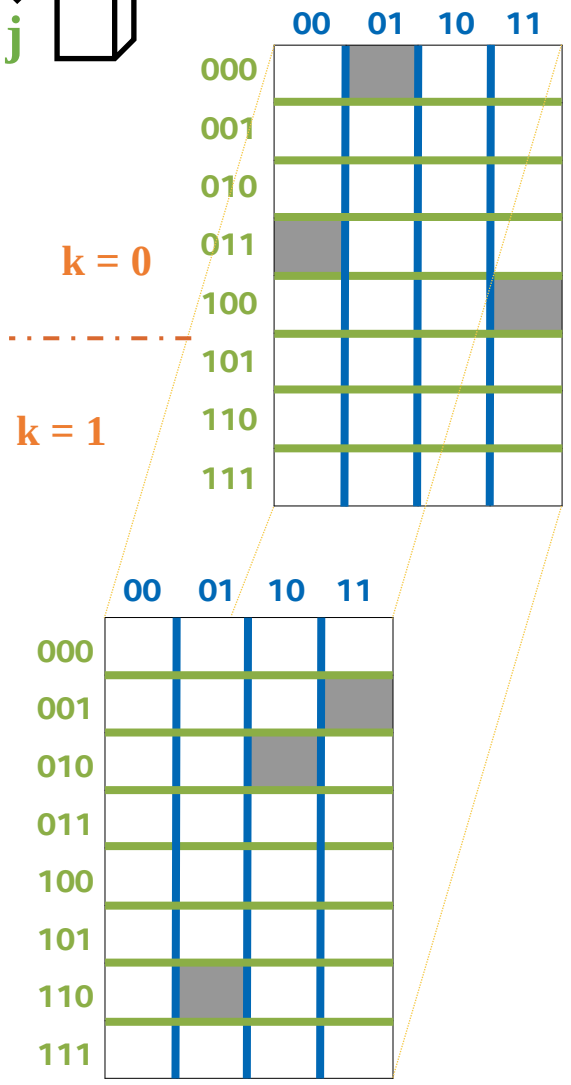




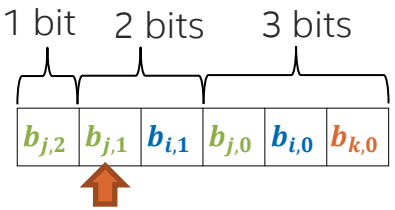
ALTO: Example

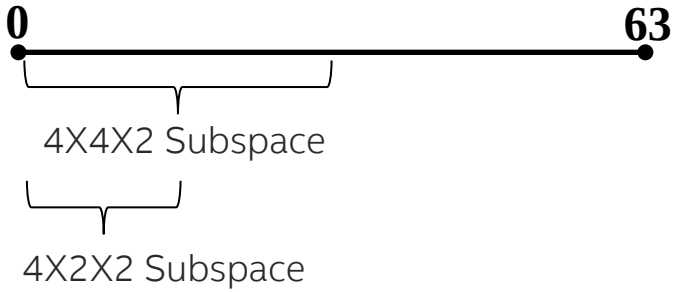


4X8X2 Tensor

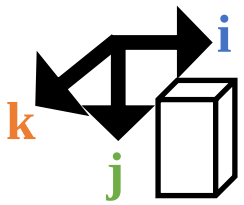


ALTO Bit Mask

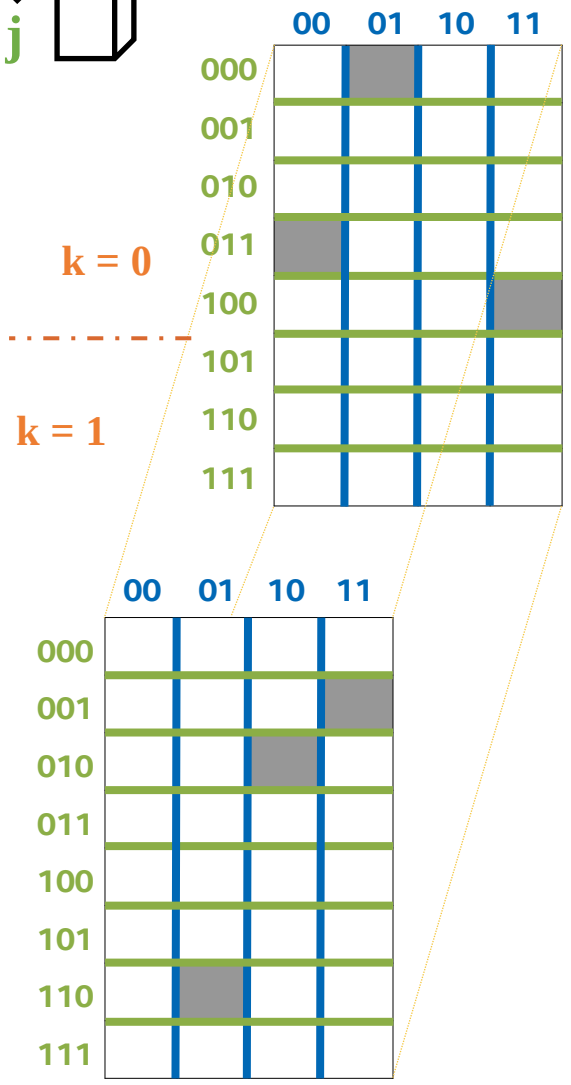




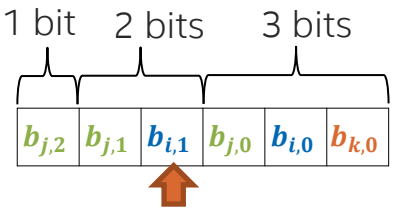
ALTO: Example

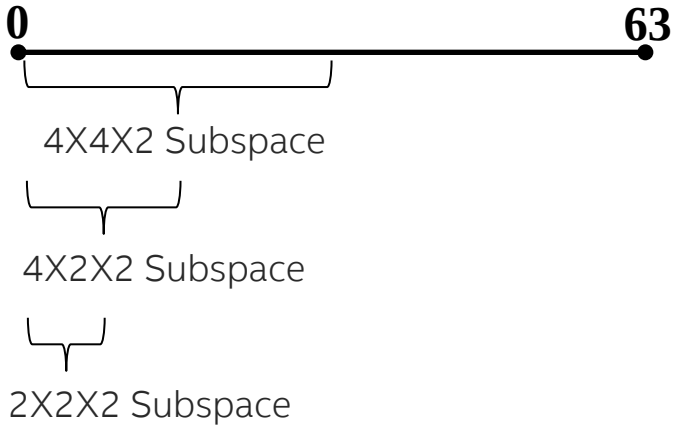


4X8X2 Tensor

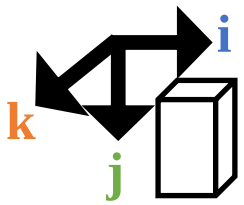


ALTO Bit Mask

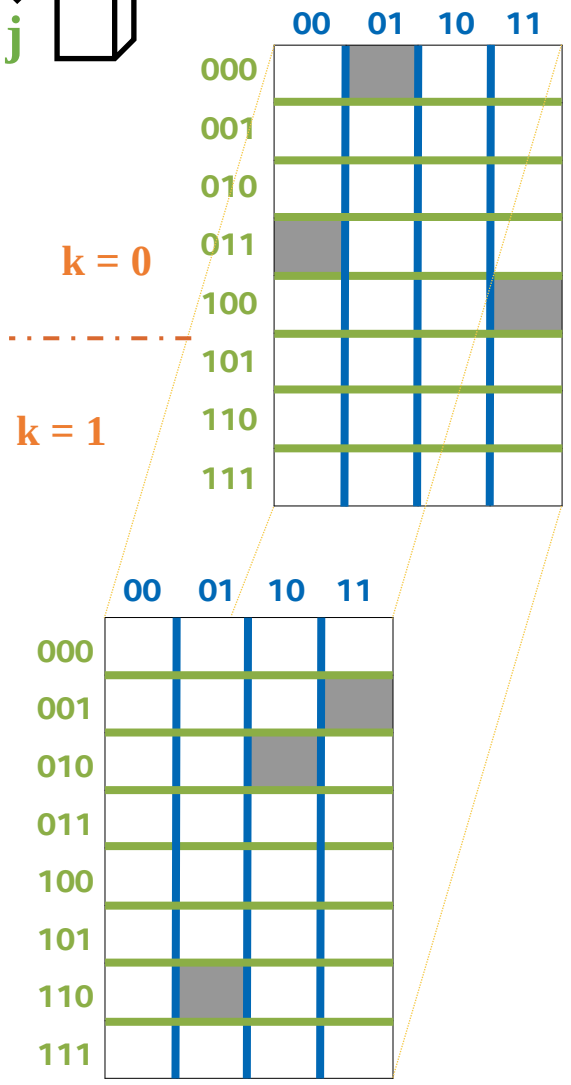




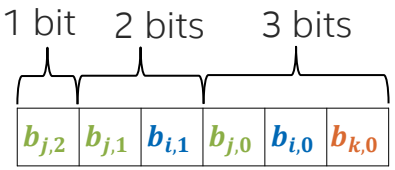
ALTO: Example



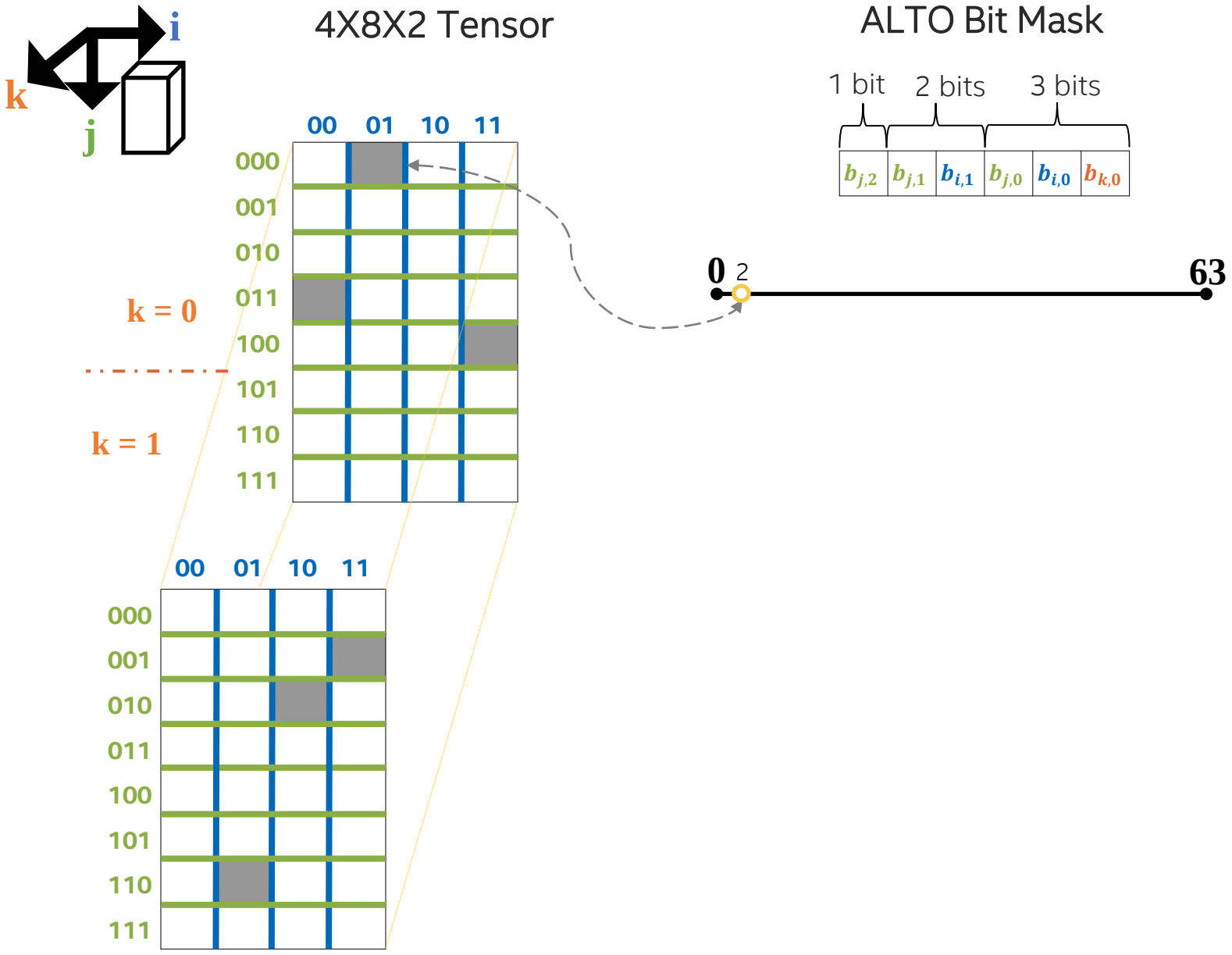
4X8X2 Tensor



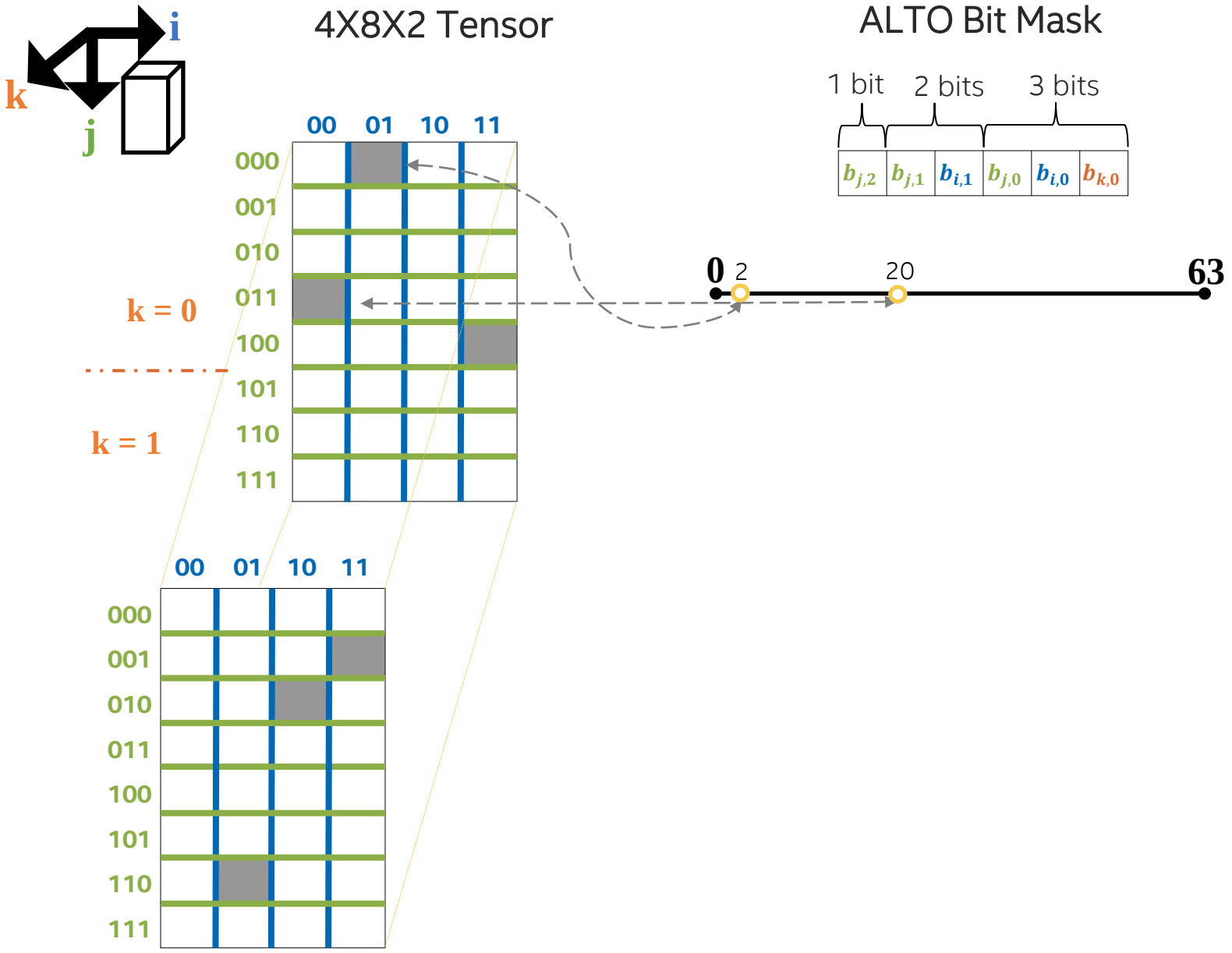
ALTO Bit Mask



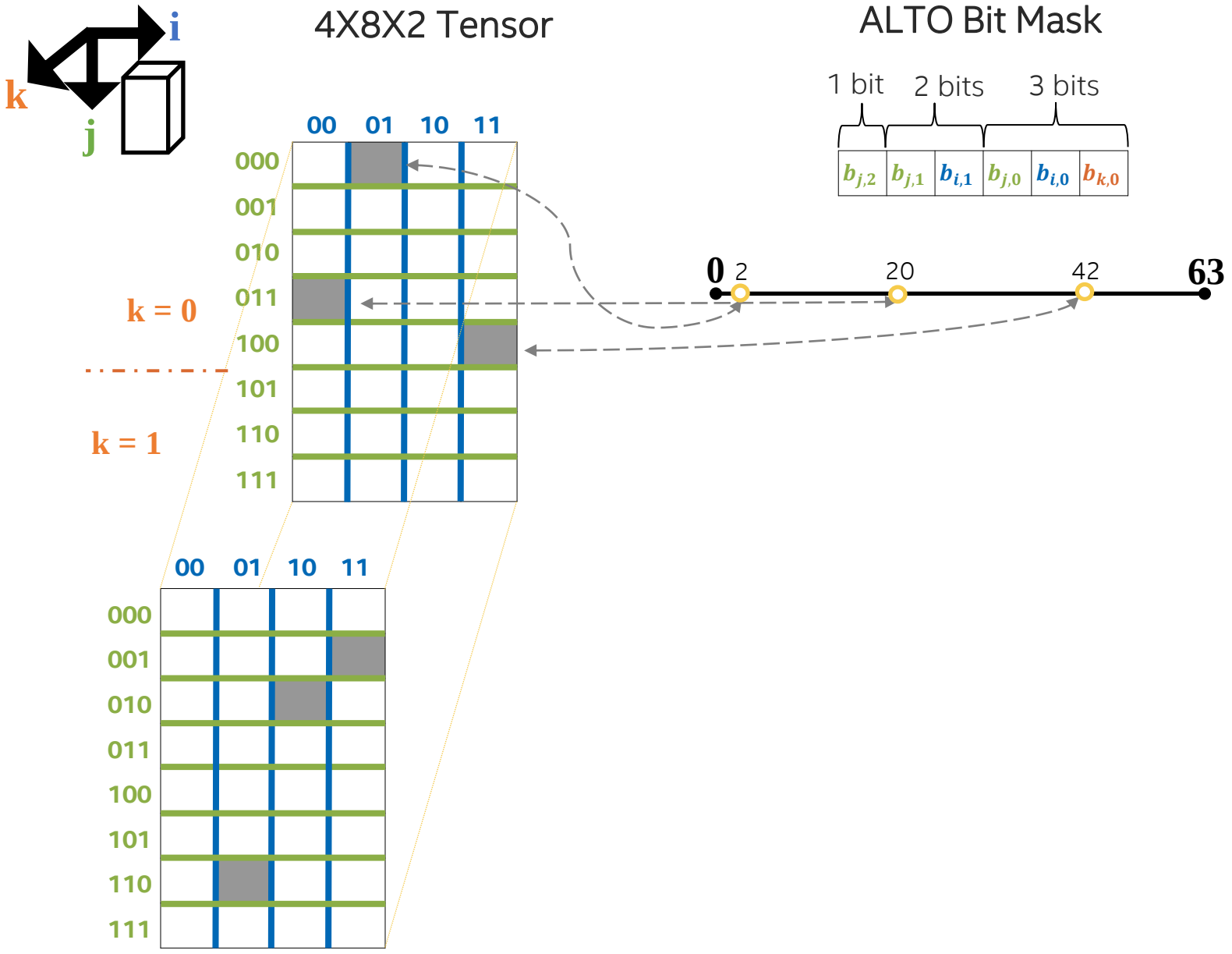
ALTO: Example



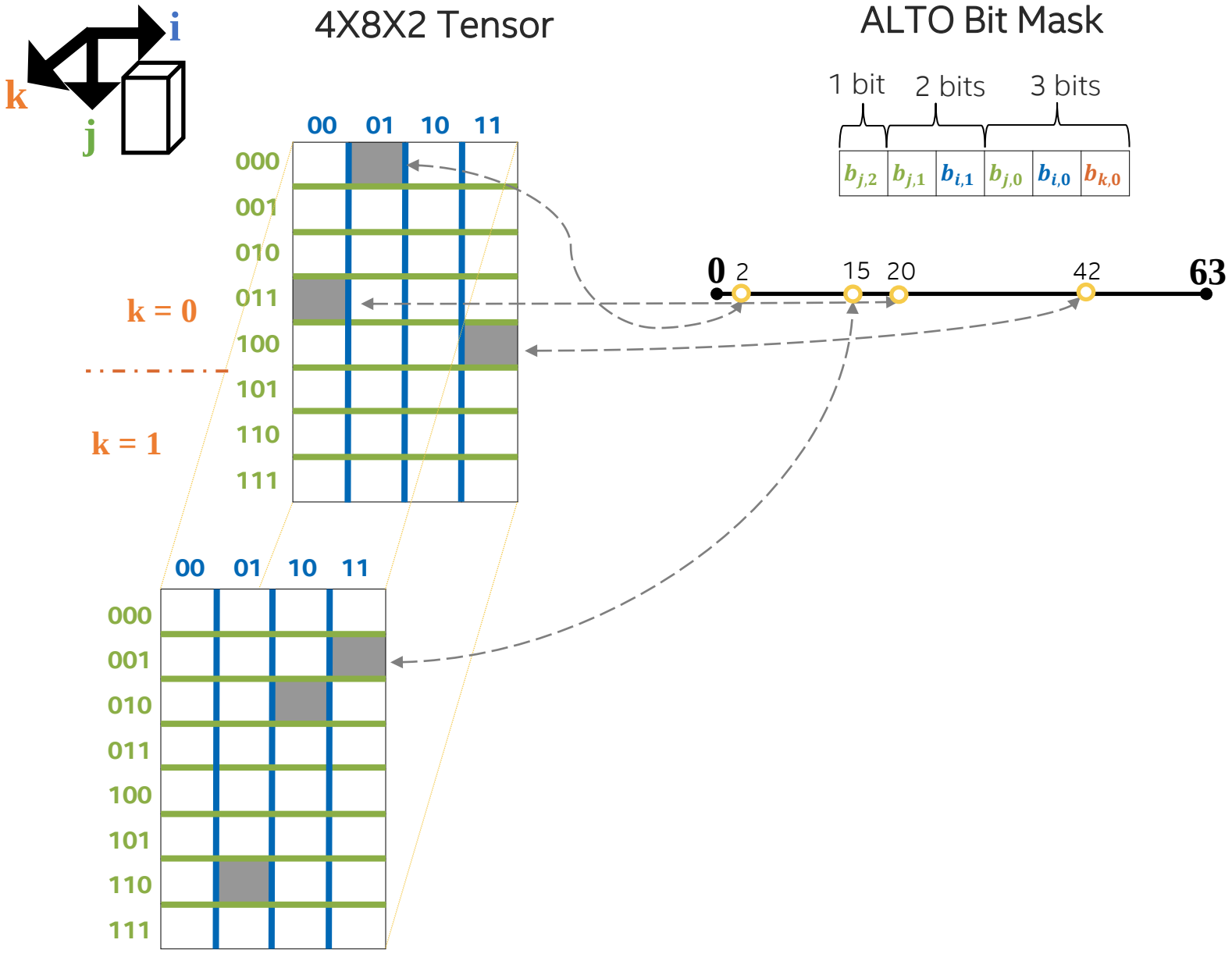
ALTO: Example



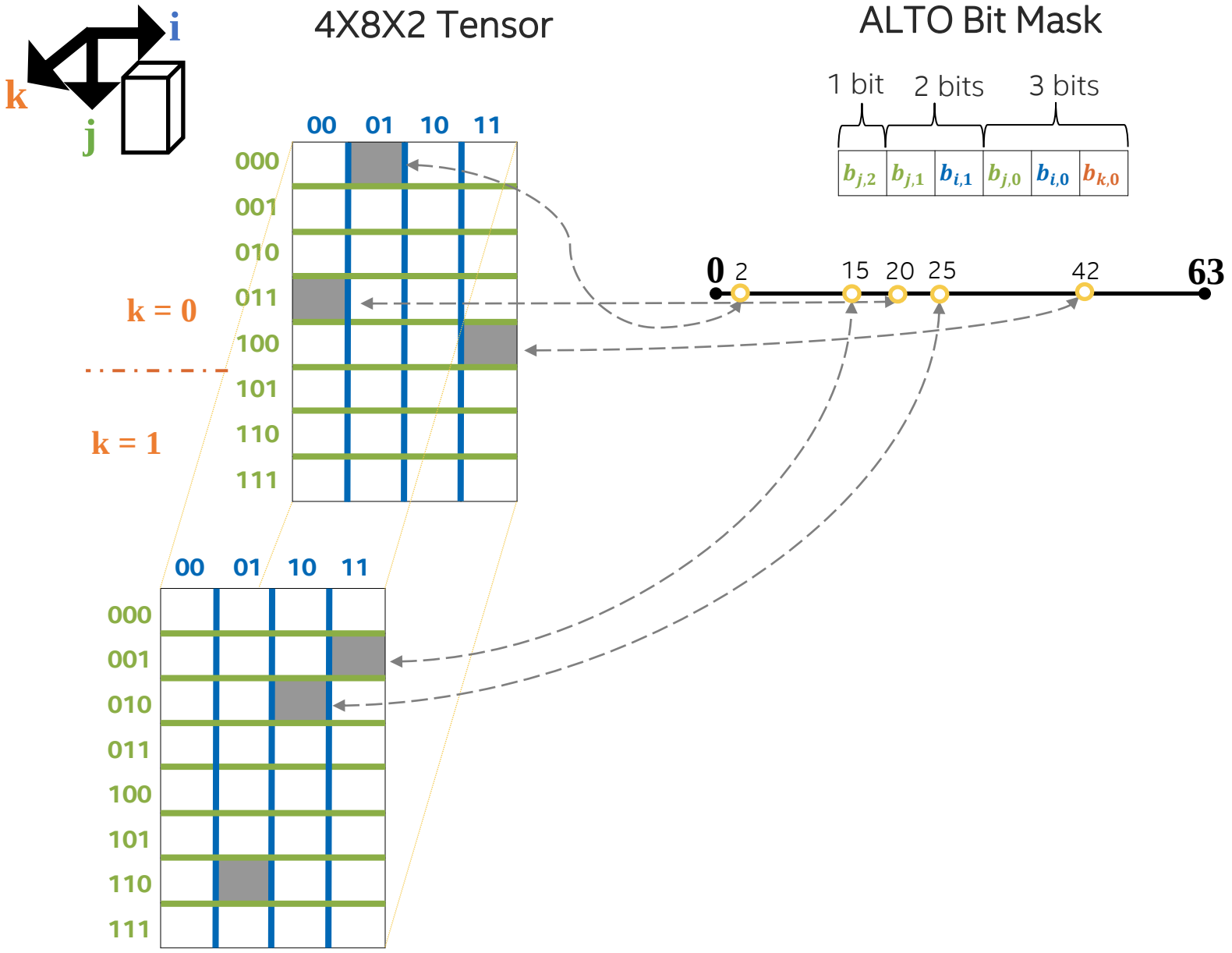
ALTO: Example



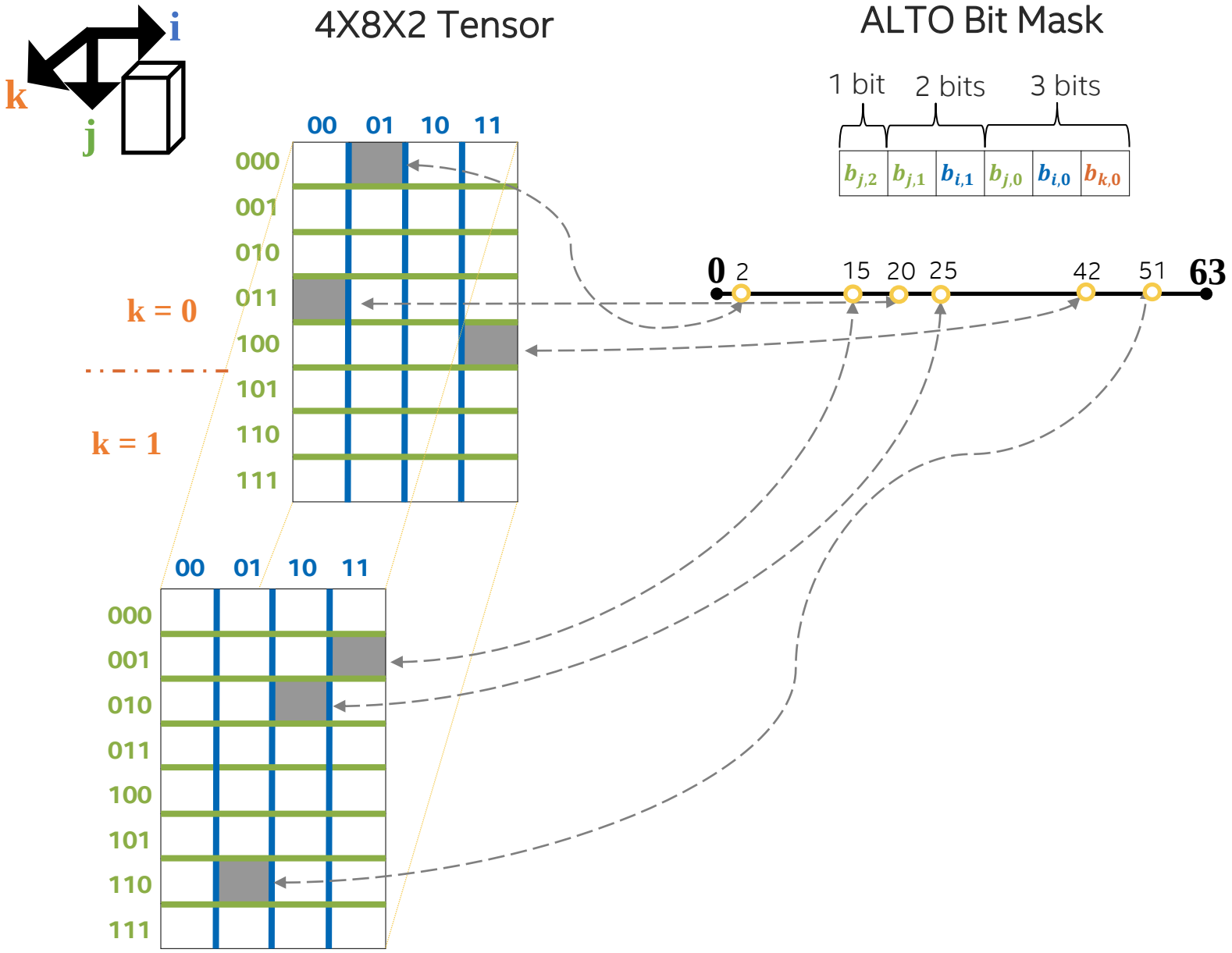
ALTO: Example



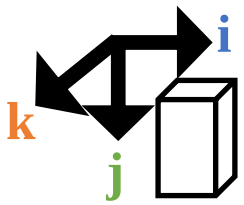
ALTO: Example



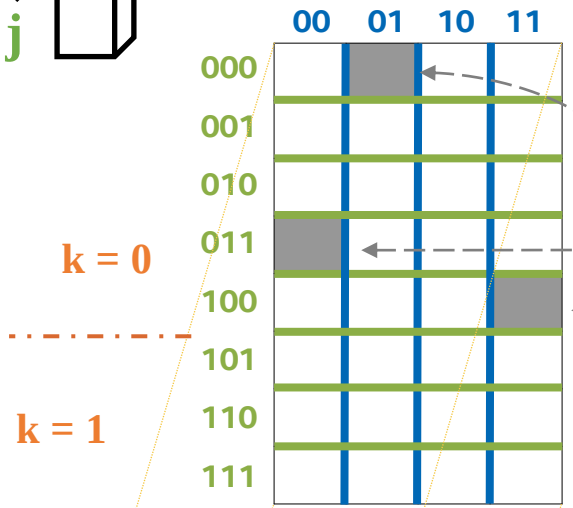
ALTO: Example

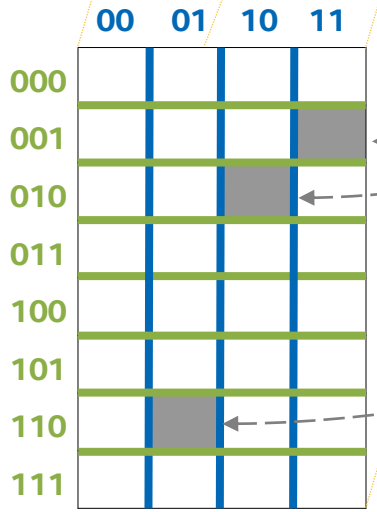


ALTO: Example

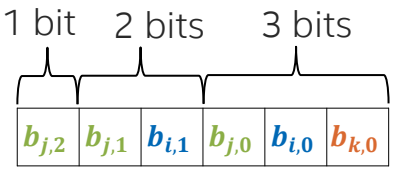


4X8X2 Tensor



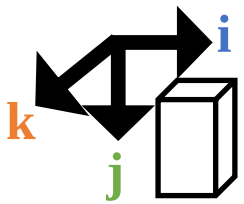


ALTO Bit Mask

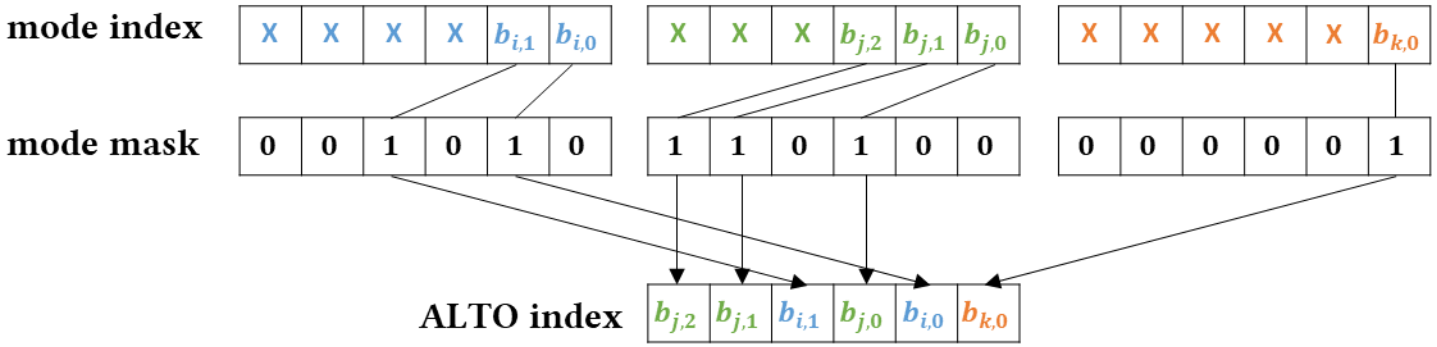


ALTO Tensor

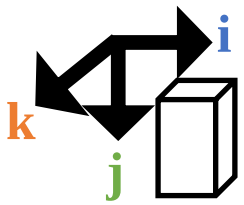
Value	Position
$x_{1,0,0}$	2 (000010)
$x_{3,1,1}$	15 (001111)
$x_{0,3,0}$	20 (010100)
$x_{2,2,1}$	25 (011001)
$x_{3,4,0}$	42 (101010)
$x_{1,6,1}$	51 (110011)



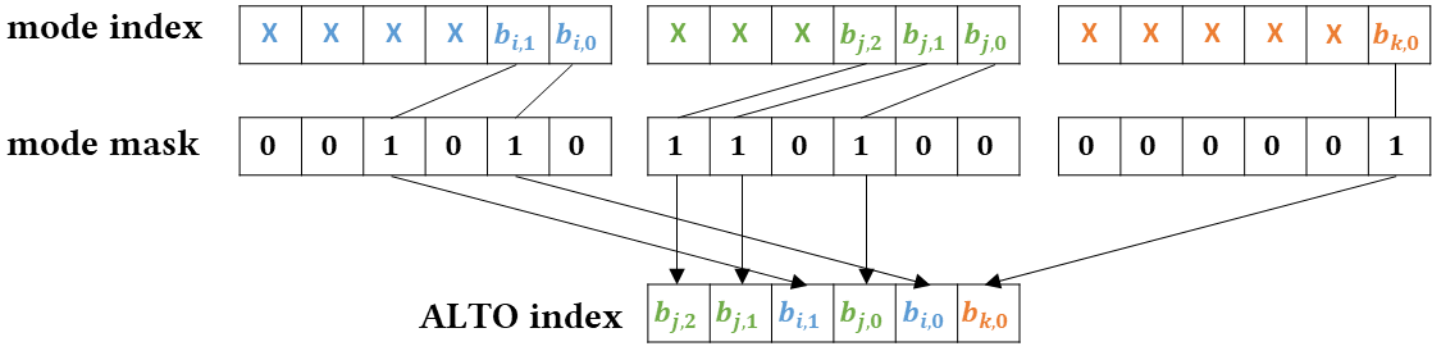
ALTO: Example



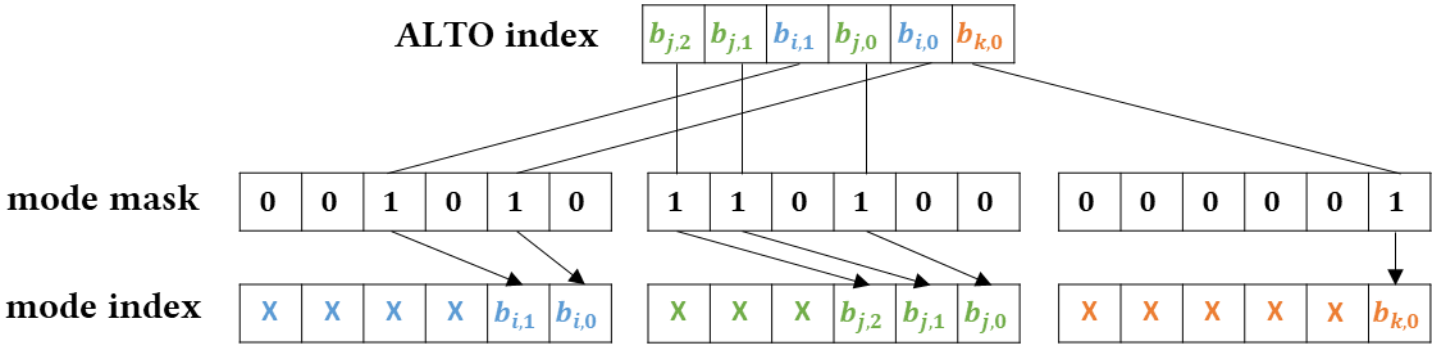
Encoding



ALTO: Example

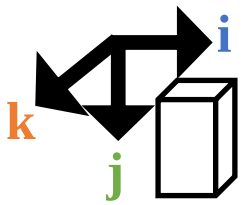


Encoding

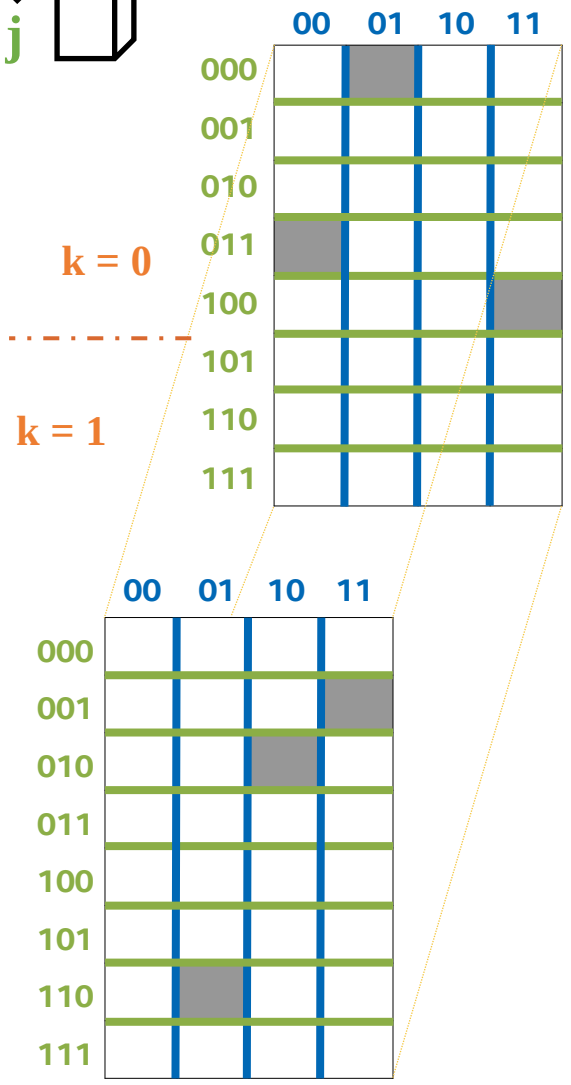


Decoding

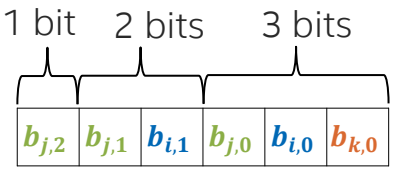
ALTO: Example



4X8X2 Tensor



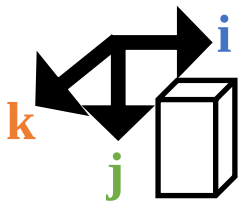
ALTO Bit Mask



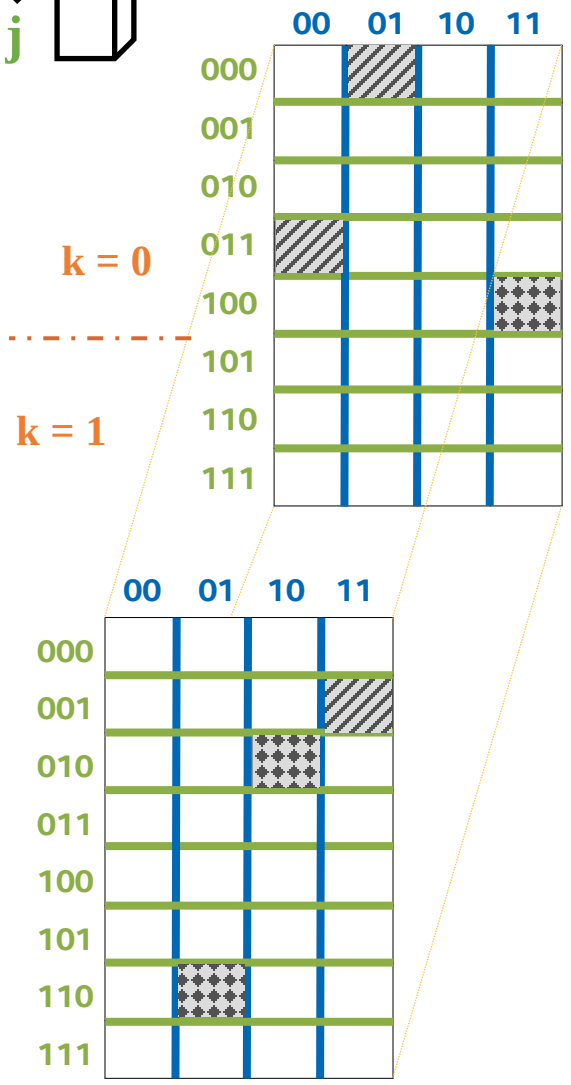
ALTO Tensor

Value	Position
$x_{1,0,0}$	2 (000010)
$x_{3,1,1}$	15 (001111)
$x_{0,3,0}$	20 (010100)
$x_{2,2,1}$	25 (011001)
$x_{3,4,0}$	42 (101010)
$x_{1,6,1}$	51 (110011)

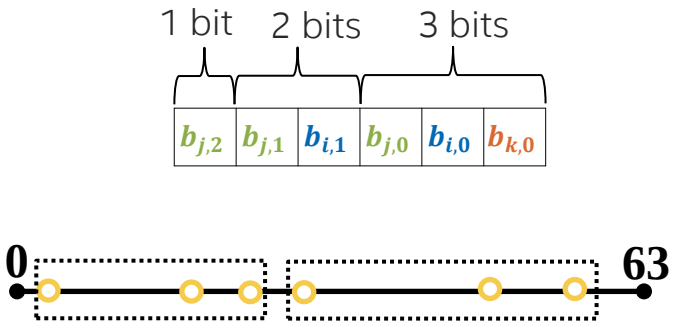
ALTO: Example



4X8X2 Tensor



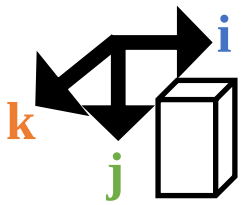
ALTO Bit Mask



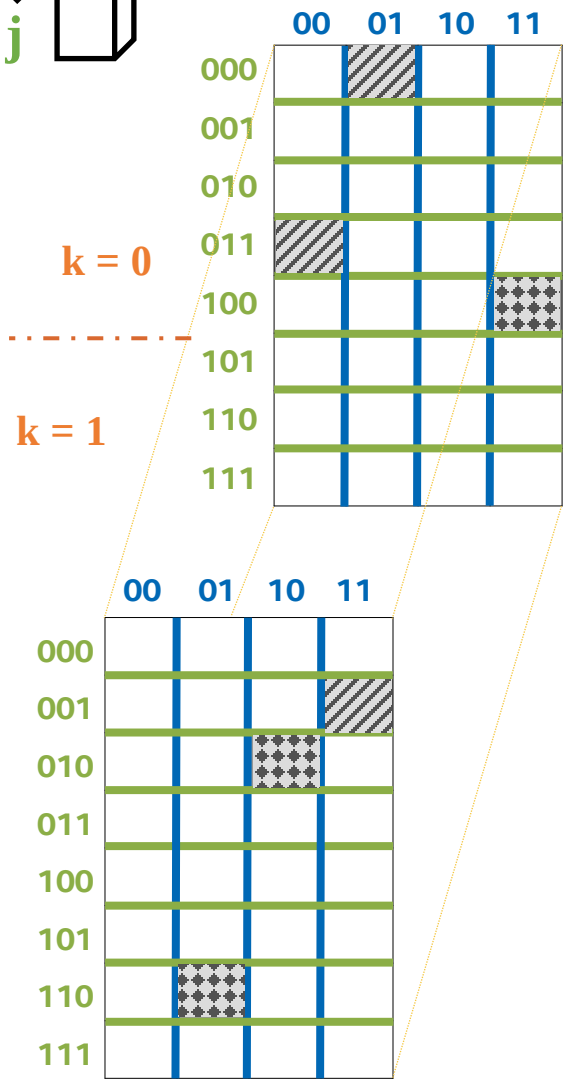
ALTO Tensor

Value	Position
$x_{1,0,0}$	2 (000010)
$x_{3,1,1}$	15 (001111)
$x_{0,3,0}$	20 (010100)
$x_{2,2,1}$	25 (011001)
$x_{3,4,0}$	42 (101010)
$x_{1,6,1}$	51 (110011)

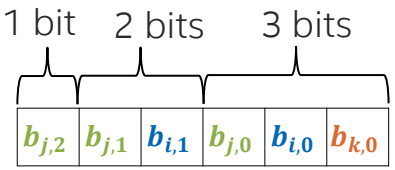
ALTO: Example

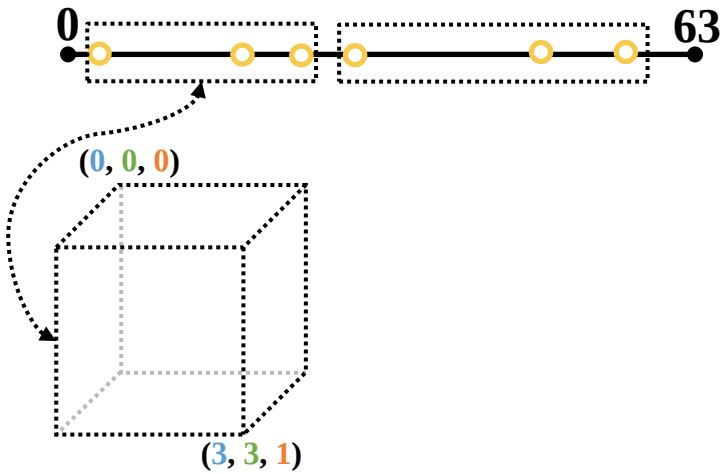


4X8X2 Tensor



ALTO Bit Mask

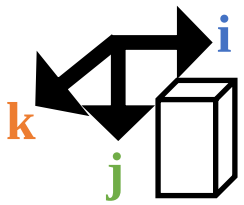




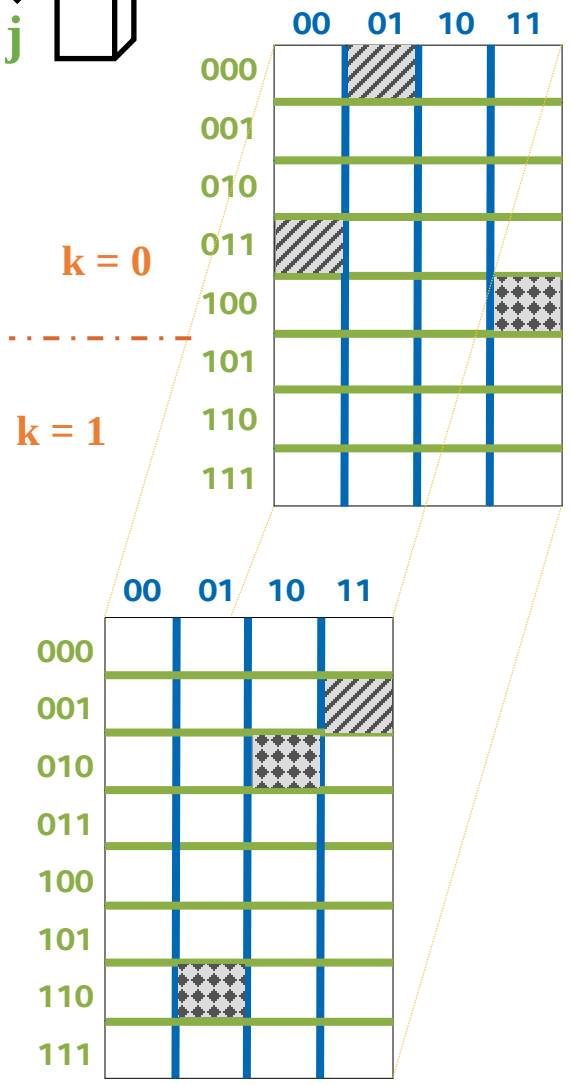
ALTO Tensor

Value	Position
$x_{1,0,0}$	2 (000010)
$x_{3,1,1}$	15 (001111)
$x_{0,3,0}$	20 (010100)
$x_{2,2,1}$	25 (011001)
$x_{3,4,0}$	42 (101010)
$x_{1,6,1}$	51 (110011)

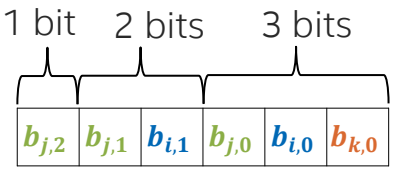
ALTO: Example

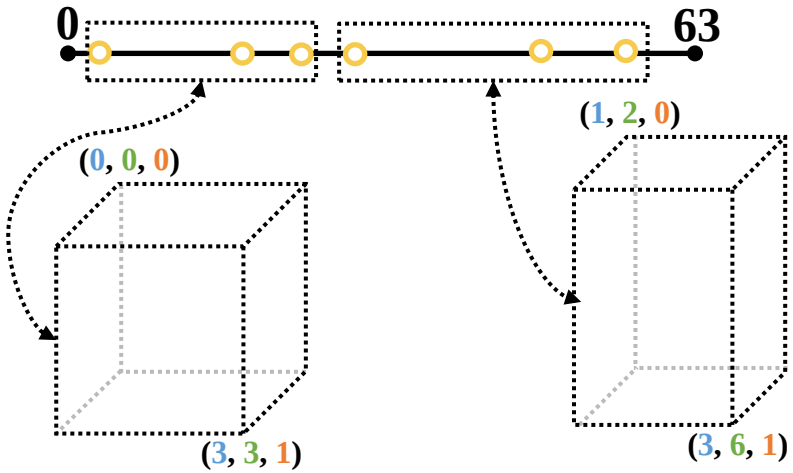


4X8X2 Tensor



ALTO Bit Mask

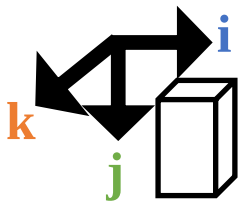




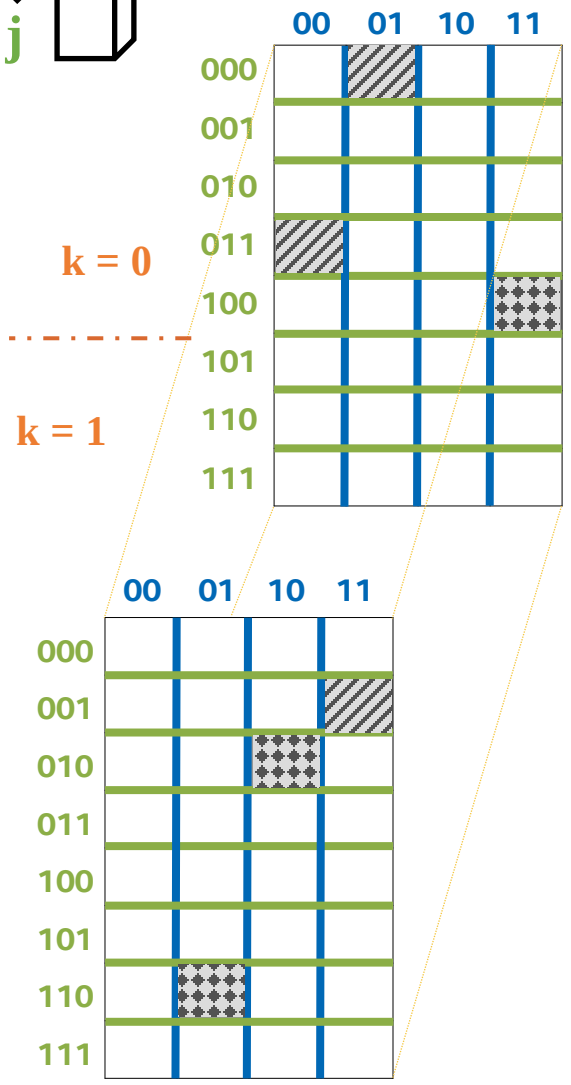
ALTO Tensor

Value	Position
$x_{1,0,0}$	2 (000010)
$x_{3,1,1}$	15 (001111)
$x_{0,3,0}$	20 (010100)
$x_{2,2,1}$	25 (011001)
$x_{3,4,0}$	42 (101010)
$x_{1,6,1}$	51 (110011)

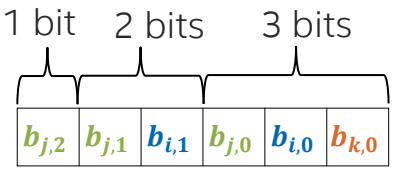
ALTO: Example

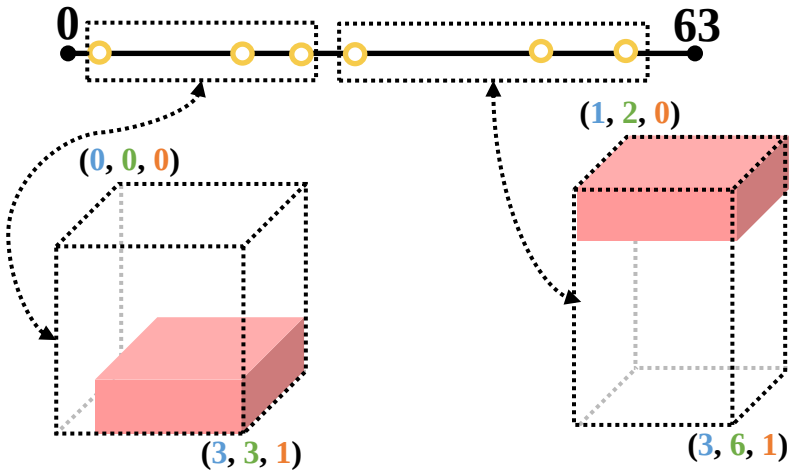


4X8X2 Tensor



ALTO Bit Mask



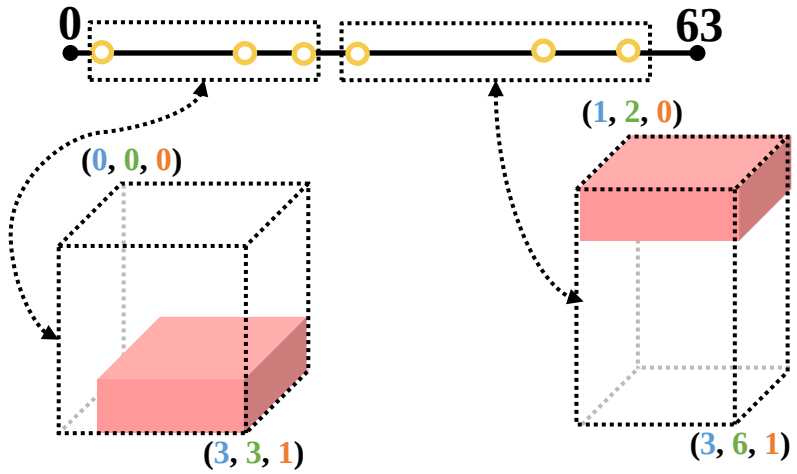


overlapping

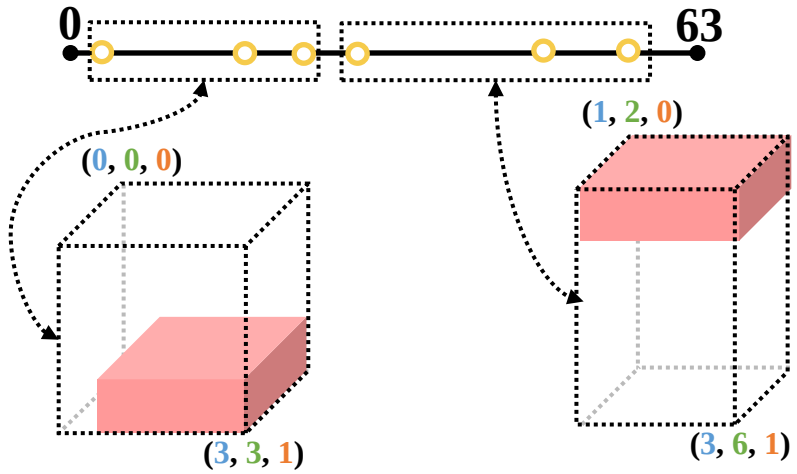
ALTO Tensor

Value	Position
$x_{1,0,0}$	2 (000010)
$x_{3,1,1}$	15 (001111)
$x_{0,3,0}$	20 (010100)
$x_{2,2,1}$	25 (011001)
$x_{3,4,0}$	42 (101010)
$x_{1,6,1}$	51 (110011)

ALTO: Adaptive conflict resolution

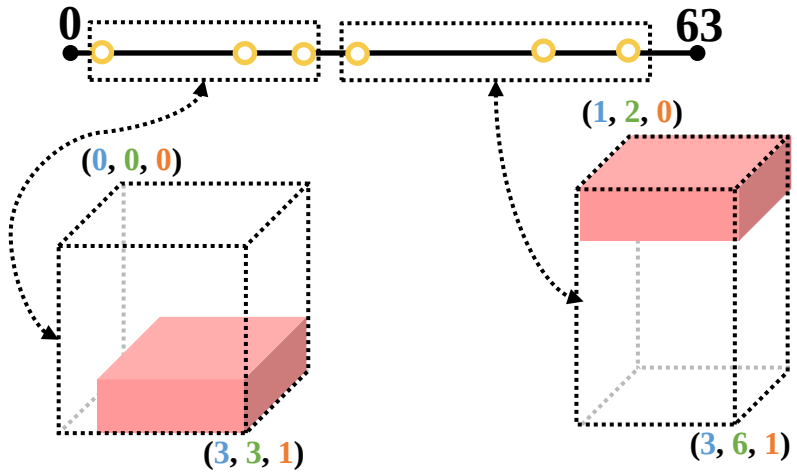


ALTO: Adaptive conflict resolution



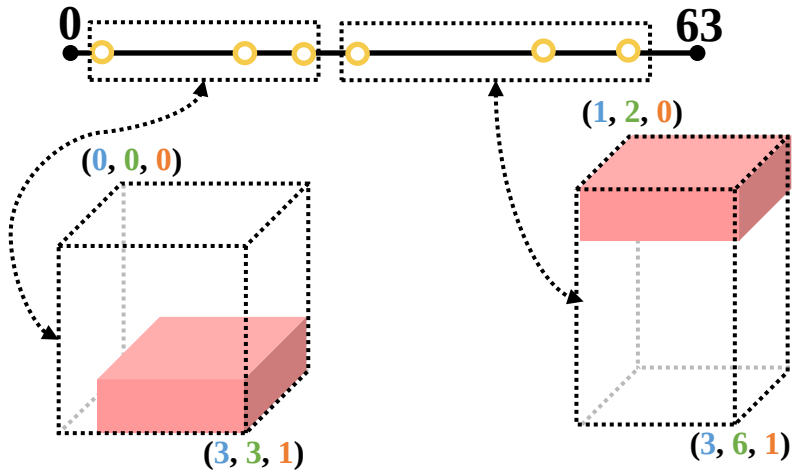
- Synchronization based on **data reuse** of target tensors

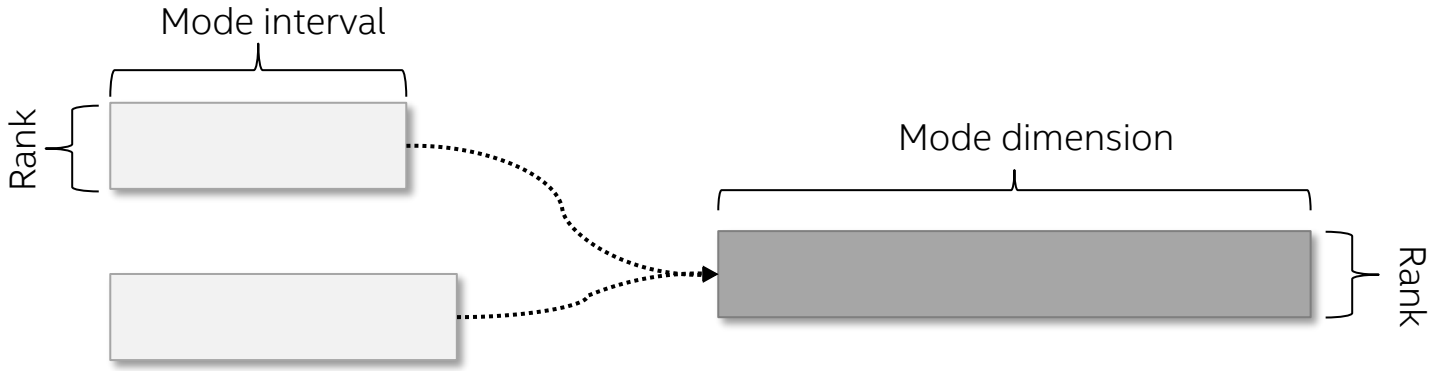
ALTO: Adaptive conflict resolution



- Synchronization based on **data reuse** of target tensors
  1. Limited or no data reuse ➔ conflict resolution via **atomic updates**

ALTO: Adaptive conflict resolution





- Synchronization based on **data reuse** of target tensors
  1. Limited or no data reuse ➔ conflict resolution via **atomic updates**
  2. Otherwise ➔ **local temporary storage** and **global parallel reduction**

ALTO: Put it all together (MTTKRP)

Algorithm Adaptive parallel execution of mode-1 MTTKRP-ALTO kernel. ALTO automatically uses local storage or atomics, based on the reuse of output fibers, to resolve the update conflicts.

Input: A third-order ALTO sparse tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ with M nonzero elements, dense factor matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$

Output: Updated dense factor matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{I \times R}$

1: **for** $l = 1, \ldots, L$ **in parallel do** ▷ ALTO line segments.
2: **for** $\forall x \in \mathcal{X}_l$ **do**
3: $i = \text{EXTRACT}(pos(x), MASK(1))$ ▷ De-linearization.
4: $j = \text{EXTRACT}(pos(x), MASK(2))$
5: $k = \text{EXTRACT}(pos(x), MASK(3))$
6: **for** $r = 1, \ldots, R$ **do**
7: $\mathbf{Temp}_l(i - T^s_{l,1}, r) \mathrel{+}= val(x) \times \mathbf{B}(j, r) \times \mathbf{C}(k, r)$
8: $\text{ATOMIC}(\tilde{\mathbf{A}}(i, r) \mathrel{+}= val(x) \times \mathbf{B}(j, r) \times \mathbf{C}(k, r))$
9: **end for**
10: **end for**
11: **end for**

ALTO: Put it all together (MTTKRP)

Algorithm Adaptive parallel execution of mode-1 MTTKRP-ALTO kernel. ALTO automatically uses local storage or atomics, based on the reuse of output fibers, to resolve the update conflicts.

Input: A third-order ALTO sparse tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ with M nonzero elements, dense factor matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$

Output: Updated dense factor matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{I \times R}$

1: **for** $l = 1, \ldots, L$ **in parallel do** ▷ ALTO line segments.
2: **for** $\forall x \in \mathcal{X}_l$ **do**
3: $i = \text{EXTRACT}(pos(x), MASK(1))$ ▷ De-linearization.
4: $j = \text{EXTRACT}(pos(x), MASK(2))$
5: $k = \text{EXTRACT}(pos(x), MASK(3))$
6: **for** $r = 1, \ldots, R$ **do**
7: $\mathbf{Temp}_l(i - T^s_{l,1}, r) \mathrel{+}= val(x) \times \mathbf{B}(j, r) \times \mathbf{C}(k, r)$
8: $\text{ATOMIC}(\tilde{\mathbf{A}}(i, r) \mathrel{+}= val(x) \times \mathbf{B}(j, r) \times \mathbf{C}(k, r))$
9: **end for**
10: **end for**
11: **end for**

ALTO: Put it all together (MTTKRP)

Algorithm Adaptive parallel execution of mode-1 MTTKRP-ALTO kernel. ALTO automatically uses local storage or atomics, based on the reuse of output fibers, to resolve the update conflicts.

Input: A third-order ALTO sparse tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ with M nonzero elements, dense factor matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$

Output: Updated dense factor matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{I \times R}$

1: **for** $l = 1, \ldots, L$ **in parallel do** ▷ ALTO line segments.
2: **for** $\forall x \in \mathcal{X}_l$ **do**
3: $i = \text{EXTRACT}(pos(x), MASK(1))$ ▷ De-linearization.
4: $j = \text{EXTRACT}(pos(x), MASK(2))$
5: $k = \text{EXTRACT}(pos(x), MASK(3))$
6: **for** $r = 1, \ldots, R$ **do**
7: $\mathbf{Temp}_l(i - T^s_{l,1}, r) \mathrel{+}= val(x) \times \mathbf{B}(j, r) \times \mathbf{C}(k, r)$
8: $\text{ATOMIC}(\tilde{\mathbf{A}}(i, r) \mathrel{+}= val(x) \times \mathbf{B}(j, r) \times \mathbf{C}(k, r))$
9: **end for**
10: **end for**
11: **end for**

ALTO: Put it all together (MTTKRP)

Algorithm Adaptive parallel execution of mode-1 MTTKRP-ALTO kernel. ALTO automatically uses local storage or atomics, based on the reuse of output fibers, to resolve the update conflicts.

Input: A third-order ALTO sparse tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ with M nonzero elements, dense factor matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$

Output: Updated dense factor matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{I \times R}$

1: **for** $l = 1, \ldots, L$ **in parallel do** ▷ ALTO line segments.
2: **for** $\forall x \in \mathcal{X}_l$ **do**
3: $i = \text{EXTRACT}(pos(x), MASK(1))$ ▷ De-linearization.
4: $j = \text{EXTRACT}(pos(x), MASK(2))$
5: $k = \text{EXTRACT}(pos(x), MASK(3))$
6: **for** $r = 1, \ldots, R$ **do**
7: $\mathbf{Temp}_l(i - T^s_{l,1}, r) \mathrel{+}= val(x) \times \mathbf{B}(j, r) \times \mathbf{C}(k, r)$
8: $\text{ATOMIC}(\tilde{\mathbf{A}}(i, r) \mathrel{+}= val(x) \times \mathbf{B}(j, r) \times \mathbf{C}(k, r))$
9: **end for**
10: **end for**
11: **end for**

ALTO: Put it all together (MTTKRP)

Algorithm Adaptive parallel execution of mode-1 MTTKRP-ALTO kernel. ALTO automatically uses local storage or atomics, based on the reuse of output fibers, to resolve the update conflicts.

Input: A third-order ALTO sparse tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ with M nonzero elements, dense factor matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$

Output: Updated dense factor matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{I \times R}$

1: **for** $l = 1, \ldots, L$ **in parallel do** ▷ ALTO line segments.
2: **for** $\forall x \in \mathcal{X}_l$ **do**
3: $i = \mathbb{EXTRACT}(pos(x), MASK(1))$ ▷ De-linearization.
4: $j = \mathbb{EXTRACT}(pos(x), MASK(2))$
5: $k = \mathbb{EXTRACT}(pos(x), MASK(3))$
6: **for** $r = 1, \ldots, R$ **do**
7: $\mathbf{Temp}_l(i - T^s_{l,1}, r) \mathrel{+}= val(x) \times \mathbf{B}(j, r) \times \mathbf{C}(k, r)$
8: $\mathbb{ATOMIC}(\tilde{\mathbf{A}}(i, r) \mathrel{+}= val(x) \times \mathbf{B}(j, r) \times \mathbf{C}(k, r))$
9: **end for**
10: **end for**
11: **end for**
12: **for** $b = 1, \ldots, I$ **in parallel do** ▷ Pull-based accumulation.
13: **for** $\forall l$ where $b \in [T^s_{l,1}, T^e_{l,1}]$ **do**
14: **for** $r = 1, \ldots, R$ **do**
15: $\tilde{\mathbf{A}}(b, r) \mathrel{+}= \mathbf{Temp}_l(b - T^s_{l,1}, r)$
16: **end for**
17: **end for**
18: **end for**
19: **return** $\tilde{\mathbf{A}}$

Results

Platform

- Cascade Lake Platinum 8280
  - 2 sockets, 28 cores each, fixed @1.8 GHz

- Intel compiler and runtime
  - “-O3 -xCORE-AVX512 -qopt-zmm-usage=high” flags to fully utilize vector units
  - Parallel runs utilize all hardware threads (112) with thread pinning via LIKWID

Results

Configurations

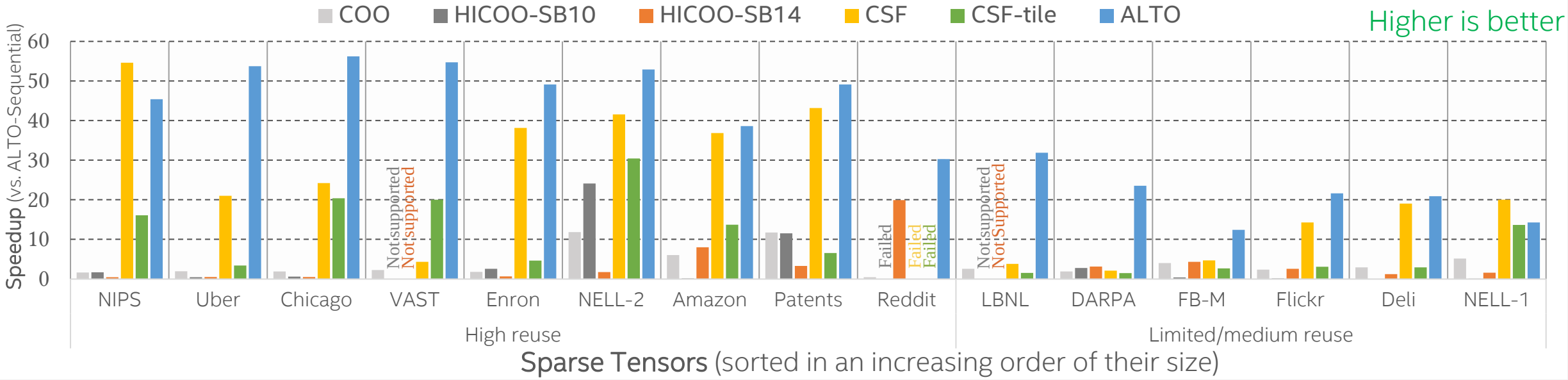
- MTTKRP with double-precision and native word (64-bit) data types
- Number of iterations = 100
- 15 tensors from FROSTT and HaTen2 datasets
  - NNZ range: [3 M, 4.7 B], Density range: [$1.5E^{-02}$, $4.3E^{-15}$]
- We consider the state-of-the-art sparse tensor libraries for CPUs
  - ParTI: coordinate (COO) and hierarchical COO (HiCOO) formats
  - SPLATT: compressed sparse fiber (CSF) and tiled CSF (CSF-tile)

Results

Parallel performance (all-modes MTTKRP)

- **47x** geomean speedup for high reuse cases (**84%** of realizable theoretical speedup)
- **22x** geomean speedup for memory-bound cases

- Prior formats' performance varies widely due to sensitivity to irregularities in shape and data

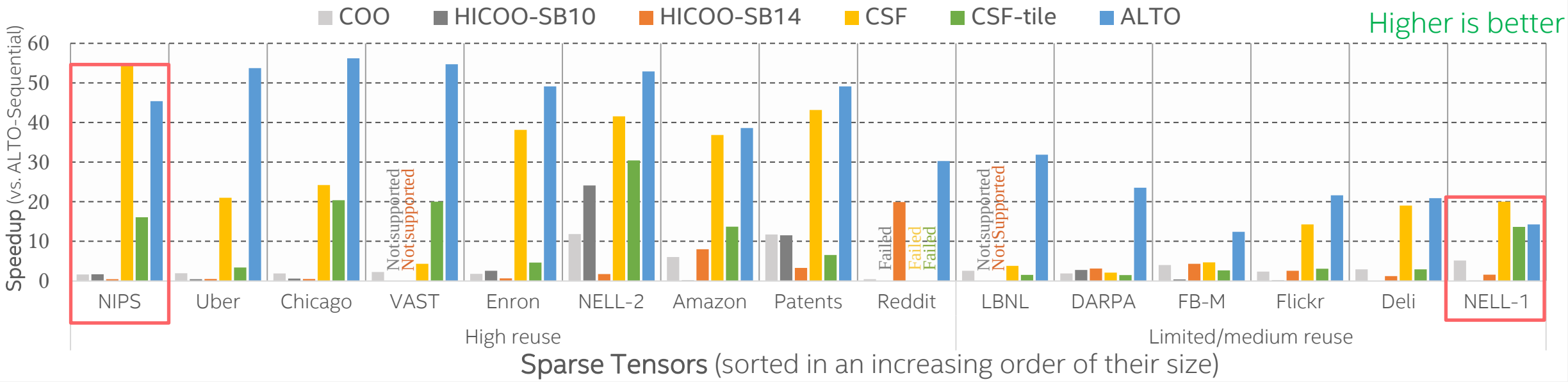


Results

Parallel performance (all-modes MTTKRP)

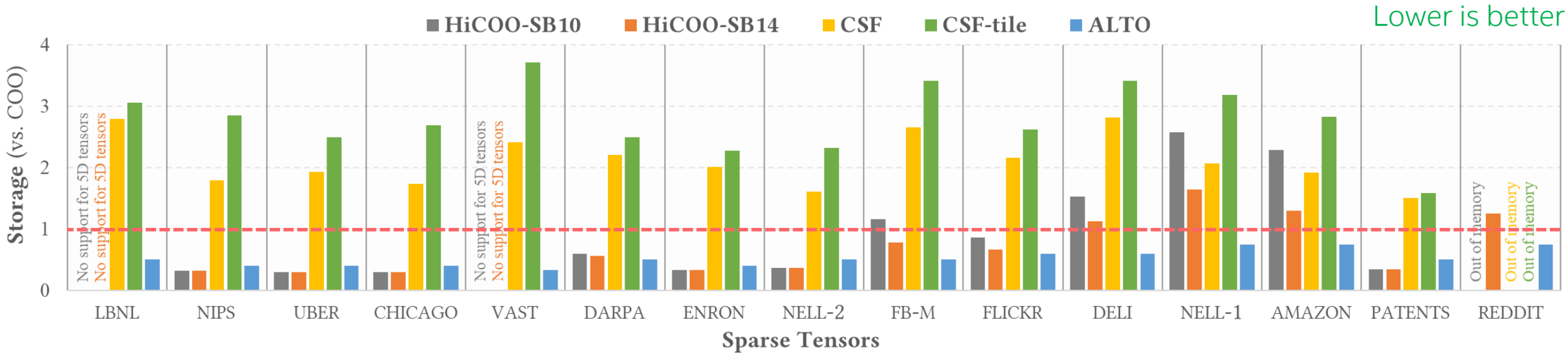
- **47x** geomean speedup for high reuse cases (**84%** of realizable theoretical speedup)
- **22x** geomean speedup for memory-bound cases

- Prior formats' performance varies widely due to sensitivity to irregularities in shape and data



Results

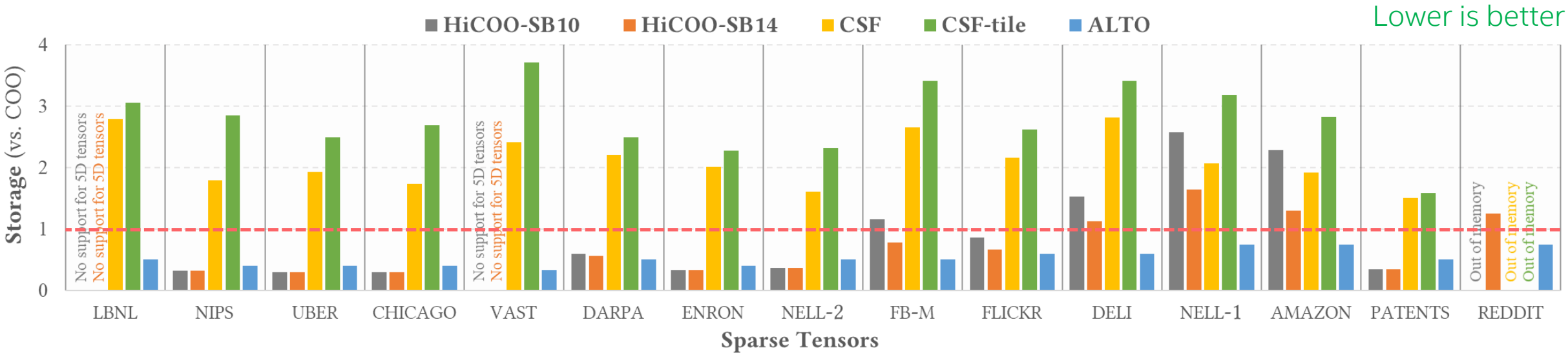
Tensor storage



Results

Tensor storage

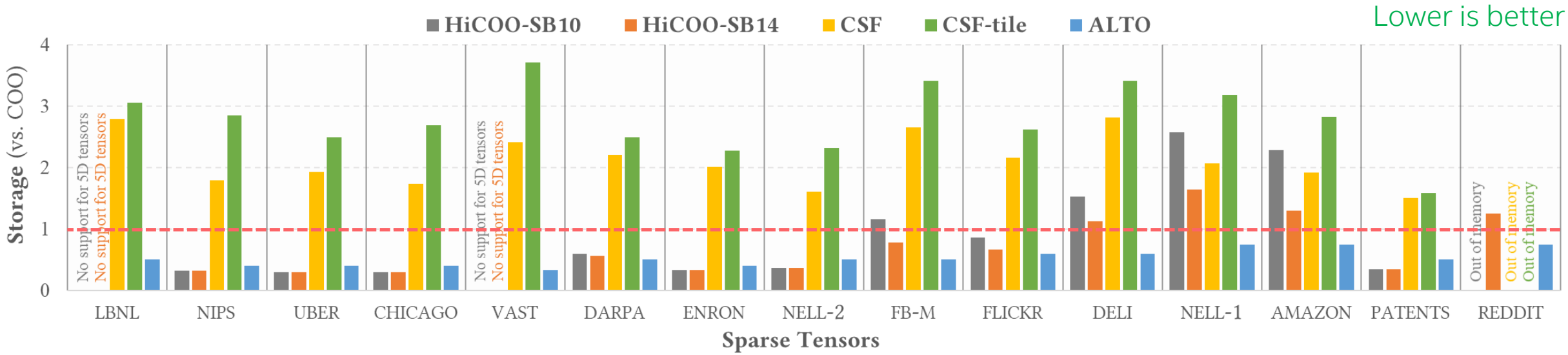
- ALTO format storage < COO



Results

Tensor storage

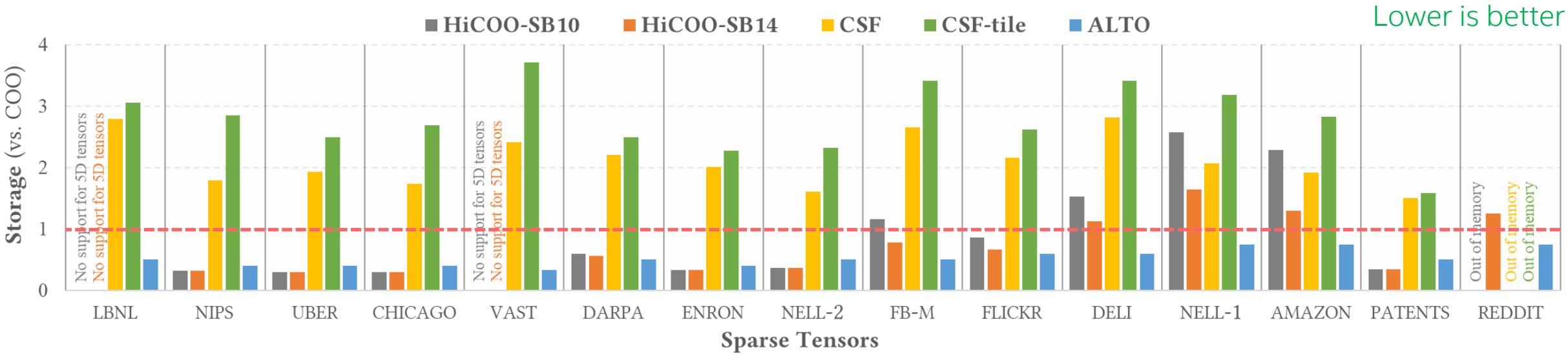
- ALTO format storage < COO
- Mode-specific formats' storage > COO
- imposing a tilling over CSF increases storage requirements



Results

Tensor storage

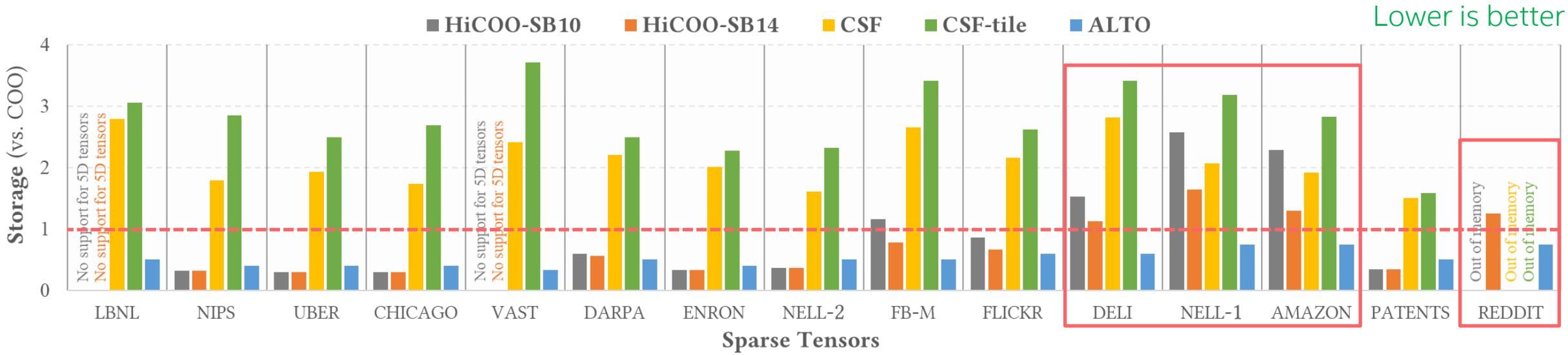
- ALTO format storage < COO
- Mode-specific formats' storage > COO
- imposing a tilling over CSF increases storage requirements
- HiCOO's memory consumption depending on
  a) block/superblock sizes
  b) spatial distribution of nonzero elements



Results

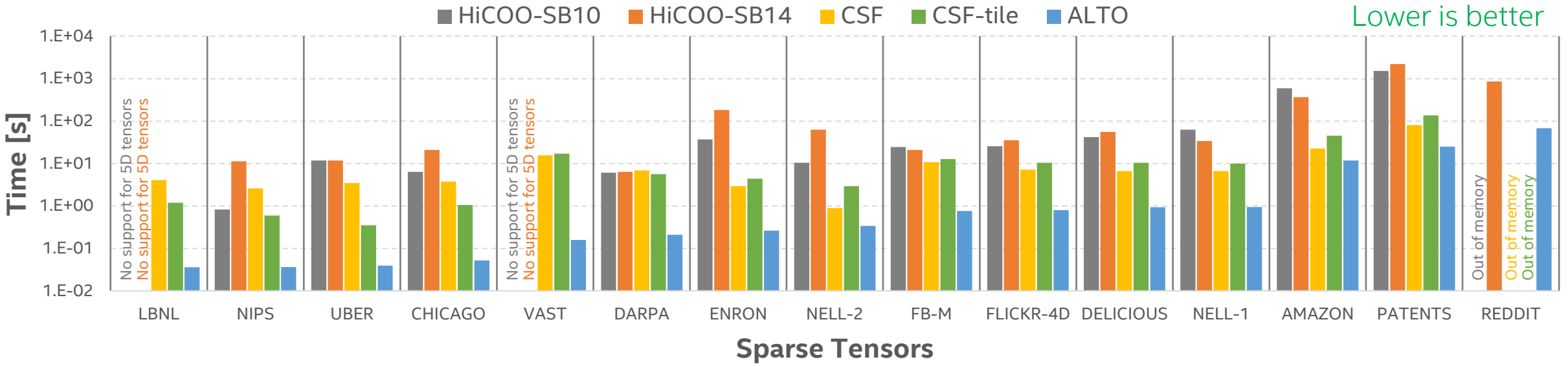
Tensor storage

- ALTO format storage < COO
- Mode-specific formats' storage > COO
- imposing a tilling over CSF increases storage requirements
- HiCOO's memory consumption depending on
  a) block/superblock sizes
  b) spatial distribution of nonzero elements



Results

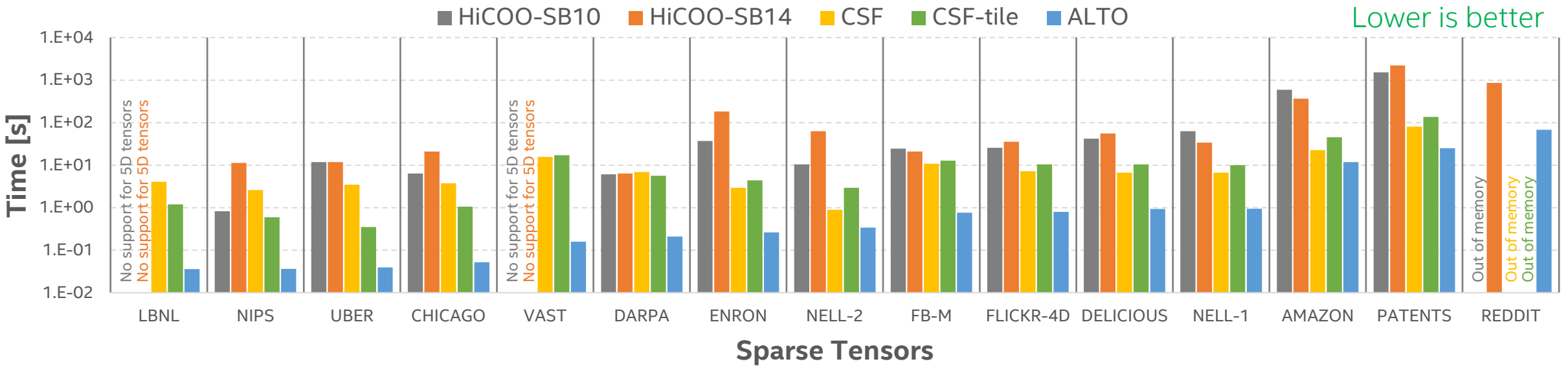
ALTO generation cost



Results

ALTO generation cost

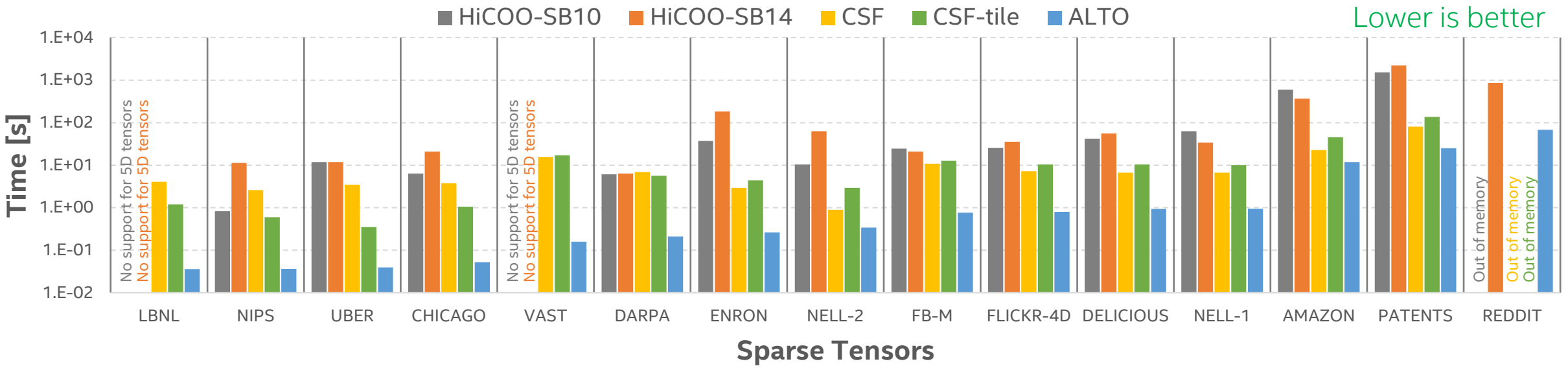
- ALTO substantially **decreases** sorting costs



Results

ALTO generation cost

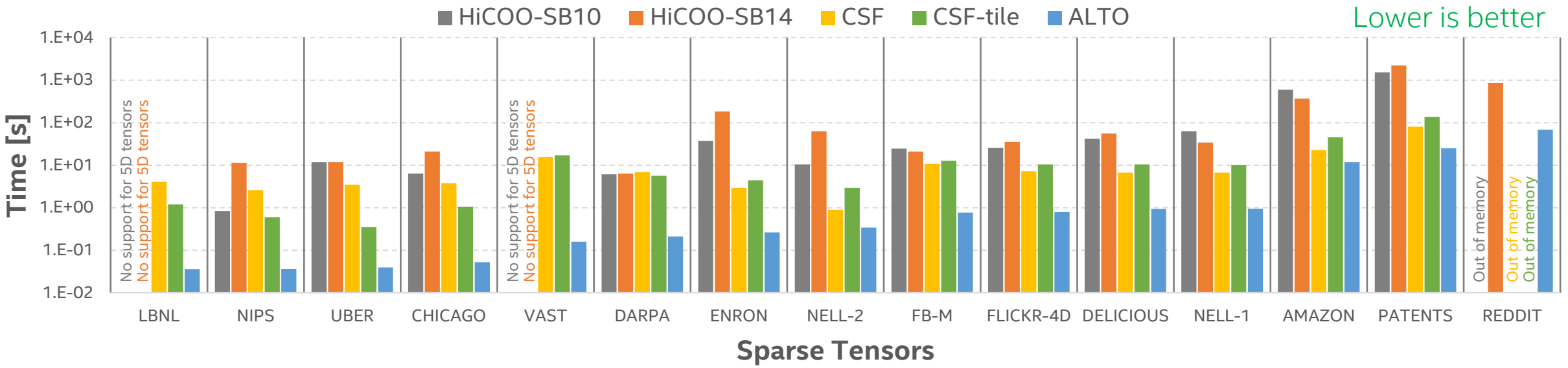
- ALTO substantially **decreases** sorting costs
- Block-based formats require expensive clustering & scheduling of NNZs



Results

ALTO generation cost

- ALTO substantially **decreases** sorting costs
- Block-based formats require expensive clustering & scheduling of NNZs
- CSF formats generated from ***presorted*** tensors; still slower than ALTO



Conclusion

- The ALTO format ***efficiently encodes*** sparse tensors of arbitrary dimensionality and spatial distributions using *a* ***single mode-agnostic*** representation

Conclusion

- The ALTO format ***efficiently encodes*** sparse tensors of arbitrary dimensionality and spatial distributions using *a* ***single mode-agnostic*** representation

- ALTO-based tensor decomposition operations proved to ***outperform*** an oracle that selects the best state-of-the-art format due to
  1. superior workload balance,
  2. adaptive synchronization,
  3. compact encoding

Conclusion

- The ALTO format ***efficiently encodes*** sparse tensors of arbitrary dimensionality and spatial distributions using *a* ***single mode-agnostic*** representation

- ALTO-based tensor decomposition operations proved to ***outperform*** an oracle that selects the best state-of-the-art format due to
  1. superior workload balance,
  2. adaptive synchronization,
  3. compact encoding

- ALTO achieves **84% parallel efficiency** in high reuse cases, 63% for all cases
  - **7.9x** (*13.2x w/ rank specialization*) **geometric-mean speedup** over the best general formats
  - **4.3x geometric-mean compression ratio** over the best mode-specific formats

Future Work

- Working on distributed-memory execution of key tensor decomposition algorithms

- Exploring other massively parallel platforms (GPGPUs)

- Investigating additional sparse tensor kernels/algorithms
  ➔ provide an **ALTO library**

ALTO pre-print: **arxiv.org/abs/2102.10245**

Backup

Tensor	Dimensions	#NNZs	Density	Fib. reuse
LBNL	$1.6K \times 4.2K \times 1.6K \times 4.2K \times 868.1K$	$1.7M$	4.2×10^{-14}	Limited
NIPS	$2.5K \times 2.9K \times 14K \times 17$	$3.1M$	1.8×10^{-06}	High
UBER	$183 \times 24 \times 1.1K \times 1.7K$	$3.3M$	3.8×10^{-04}	High
CHICAGO	$6.2K \times 24 \times 77 \times 32$	$5.3M$	1.5×10^{-02}	High
VAST	$165.4K \times 11.4K \times 2 \times 100 \times 89$	$26M$	7.8×10^{-07}	High
DARPA	$22.5K \times 22.5K \times 23.8M$	$28.4M$	2.4×10^{-09}	Limited
ENRON	$6K \times 5.7K \times 244.3K \times 1.2K$	$54.2M$	5.5×10^{-09}	High
NELL-2	$12.1K \times 9.2K \times 28.8K$	$76.9M$	2.4×10^{-05}	High
FB-M	$23.3M \times 23.3M \times 166$	$99.6M$	1.1×10^{-09}	Limited
FLICKR	$319.7K \times 28.2M \times 1.6M \times 731$	$112.9M$	1.1×10^{-14}	Limited
DELI	$532.9K \times 17.3M \times 2.5M \times 1.4K$	$140.1M$	4.3×10^{-15}	Medium
NELL-1	$2.9M \times 2.1M \times 25.5M$	$143.6M$	9.1×10^{-13}	Medium
AMAZON	$4.8M \times 1.8M \times 1.8M$	$1.7B$	1.1×10^{-10}	High
PATENTS	$46 \times 239.2K \times 239.2K$	$3.6B$	1.4×10^{-03}	High
REDDIT	$8.2M \times 177K \times 8.1M$	$4.7B$	4.0×10^{-10}	High