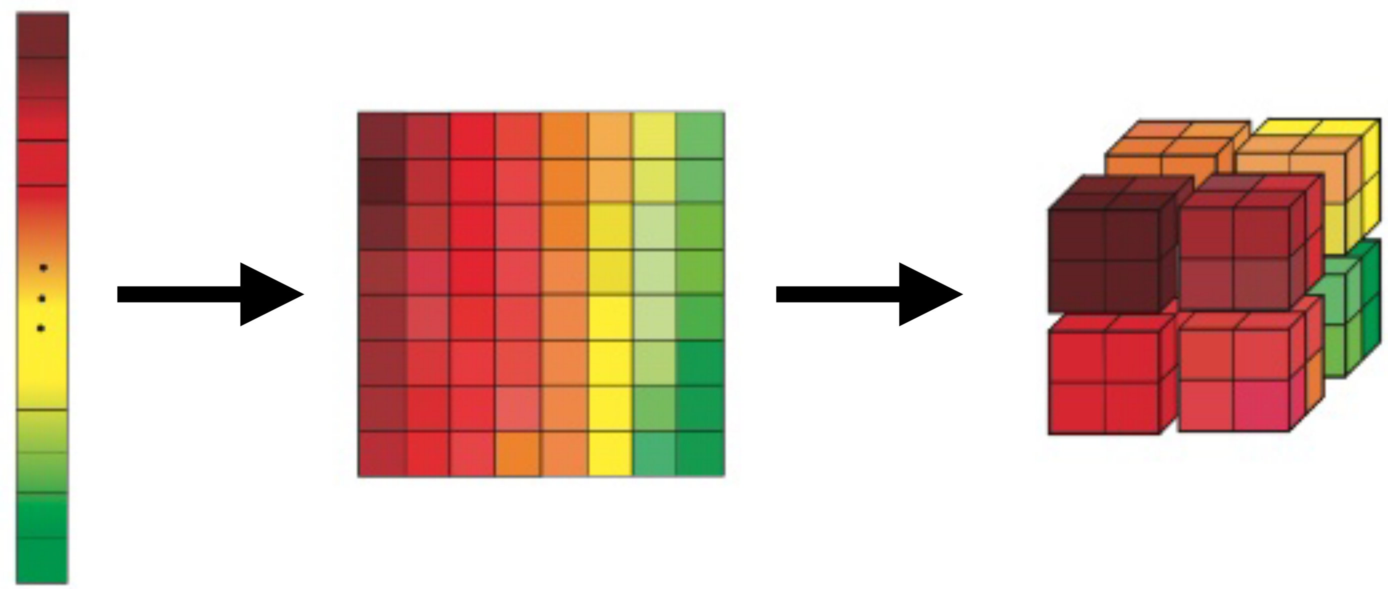


Parallel Sparse Tensor Decompositions using HiCOO Format

Jiajia Li, Jee Choi, Richard Vuduc

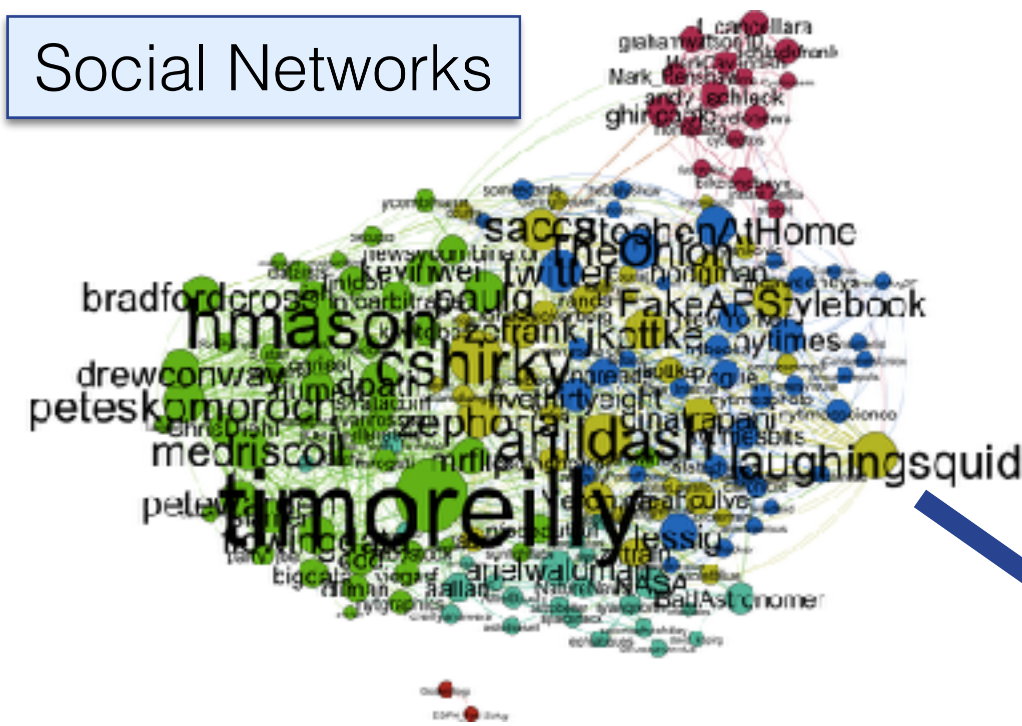
May 08, 2018 @ SIAM ALA'18

Outline

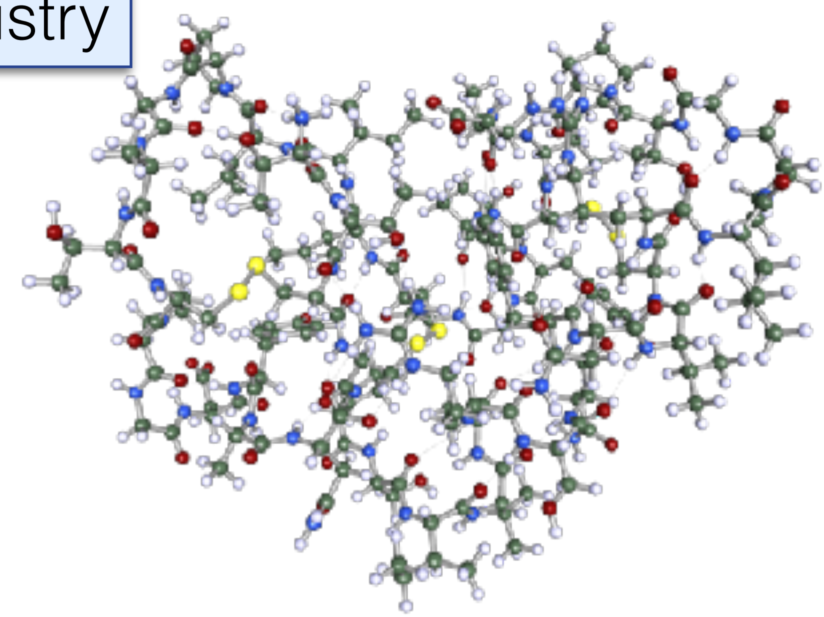
- Background
- HiCOO Format
- Multicore CP-ALS
- Distributed CPDs

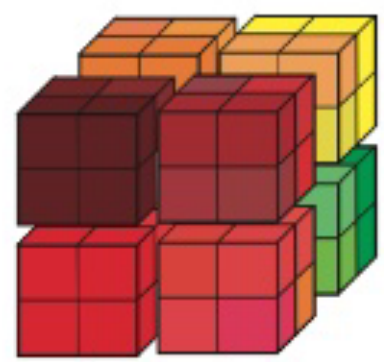
Tensors

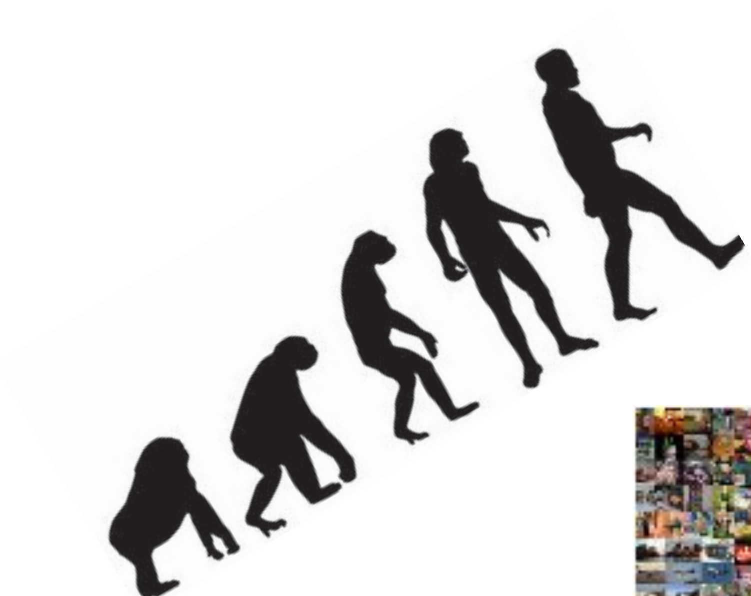
Social Networks

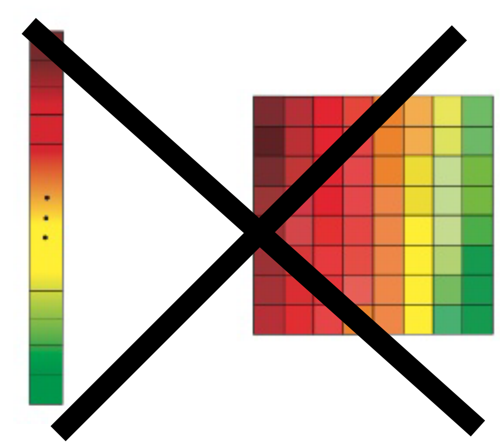


Quantum Chemistry







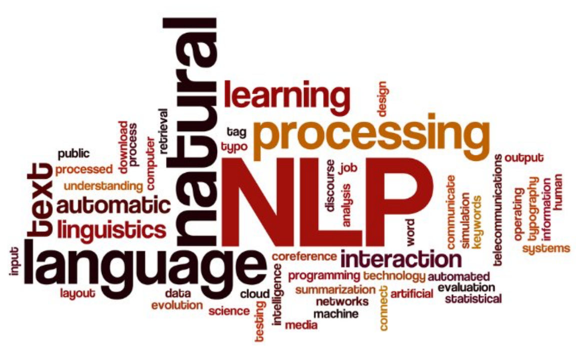


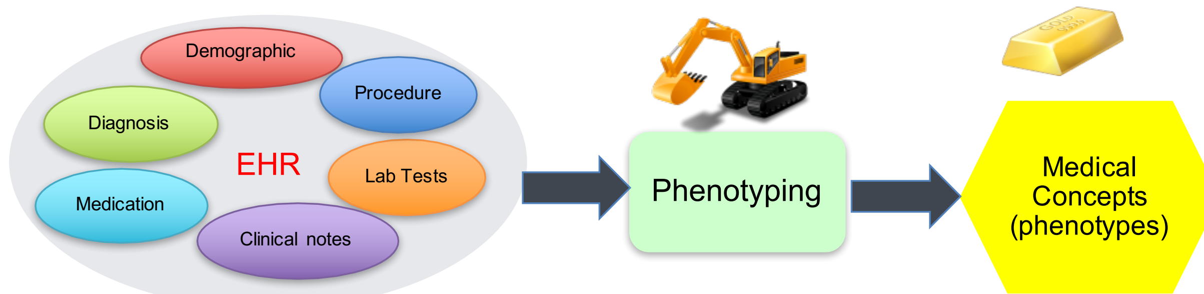
Deep Learning

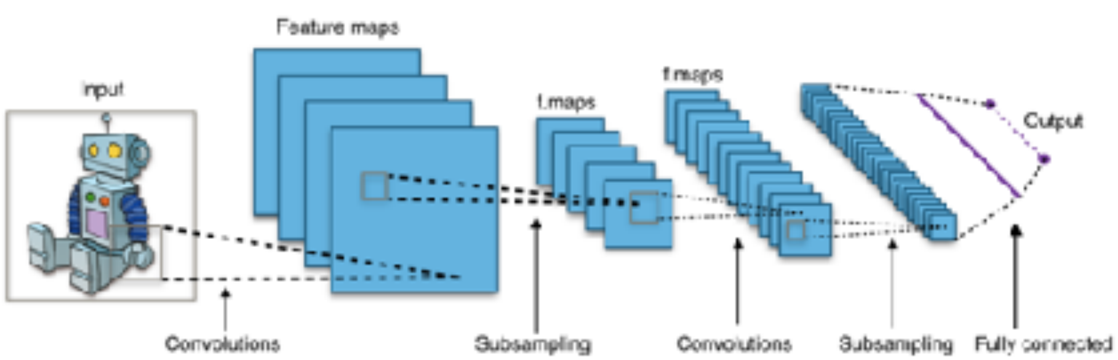
NOT ENOUGH

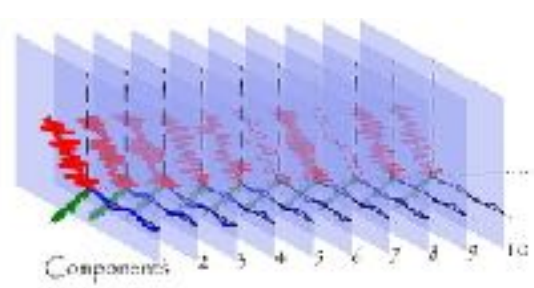
Applications of Tensor Methods

- Data analytics and compression
  - Natural language processing, healthcare analytics, social network analytics, brain signal processing, and deep learning
- Scientific computing
  - Quantum chemistry, computational physics, quantum mechanics



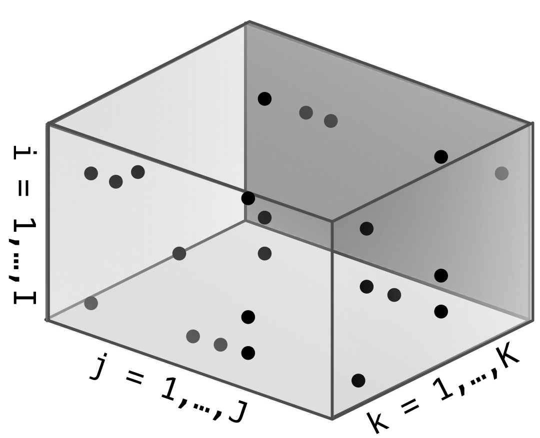






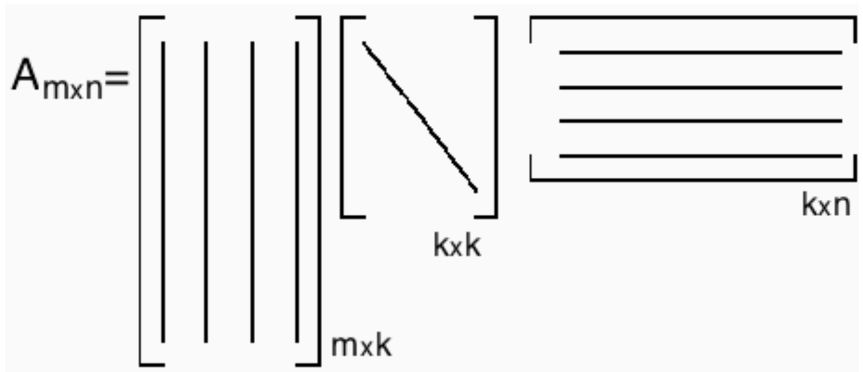
Tensors & Decompositions

- Tensors, multi-way arrays, provide a natural way to represent multidimensional data.
  - Special cases: matrices - 2D tensors, vectors - 1D tensors.
  - Tensor mode or order: tensor dimension.
  - A SPARSE tensor, a tensor consisting mostly of zero entries, widely exists in real applications.



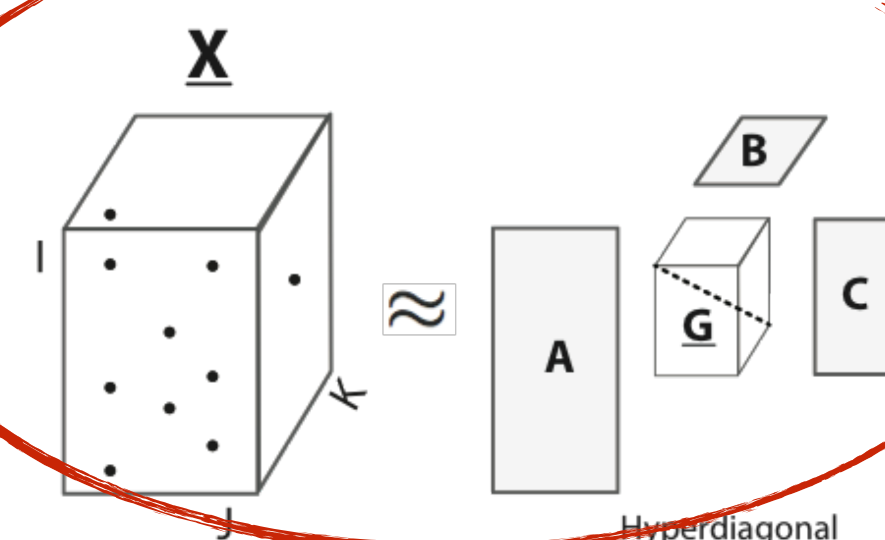
- Tensor decomposition: an extension of matrix factorization to analyze tensor features.

Singular Value Decomposition (SVD)



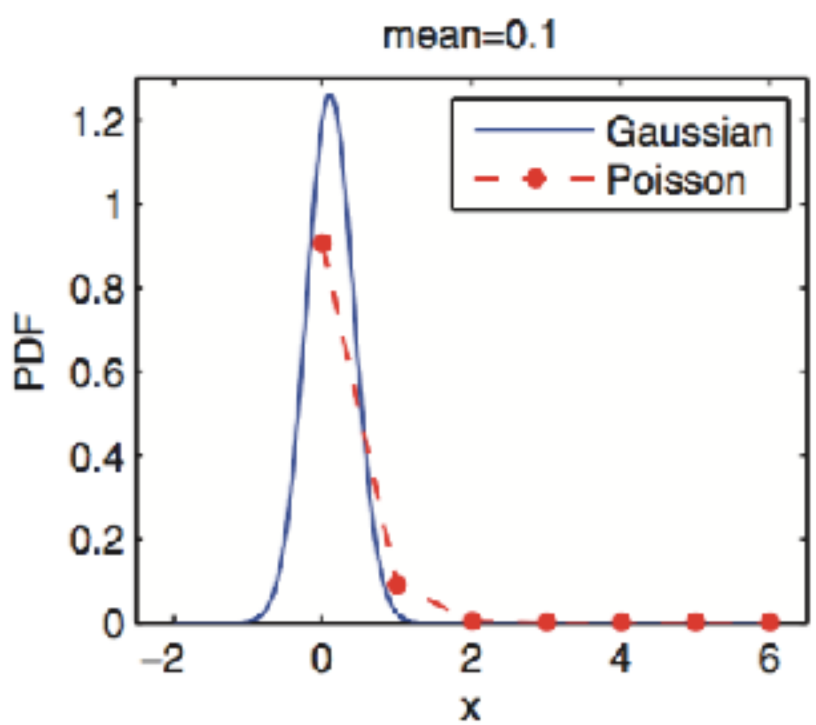
CANDECOMP/PARAFAC

CP Decomposition



CP-ALS and CP-APR

- CP-ALS - alternating least squares
  - For general data
  - Describe input data as Gaussian distribution
  - Fitting function: least squares.
  - Core operation: MTTKRP



- CP-APR – alternating Poisson regression
  - For non-negative data, e.g. count data.
  - Describe input data as Poisson distribution
  - Fitting function:Poisson likelihood fitting algorithm
  - Core operation: element-wise division.

CP-ALS

Bottleneck: MTTKRP

procedure CP-ALS($\mathfrak{X}$,R)
  initialize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$
  repeat
    for $n = 1, \ldots, N$ **do**
      $\mathbf{V} \leftarrow \mathbf{A}^{(1)\mathsf{T}}\mathbf{A}^{(1)} * \cdots * \mathbf{A}^{(n-1)\mathsf{T}}\mathbf{A}^{(n-1)} * \mathbf{A}^{(n+1)\mathsf{T}}\mathbf{A}^{(n+1)} * \cdots * \mathbf{A}^{(N)\mathsf{T}}\mathbf{A}^{(N)}$
      $\mathbf{A}^{(n)} \leftarrow \mathbf{X}_{(n)}(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)})\mathbf{V}^{\dagger}$
      normalize columns of $\mathbf{A}^{(n)}$ (storing norms as $\boldsymbol{\lambda}$)
    end for
  until fit ceases to improve or maximum iterations exhausted
  return $\boldsymbol{\lambda}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$
end procedure

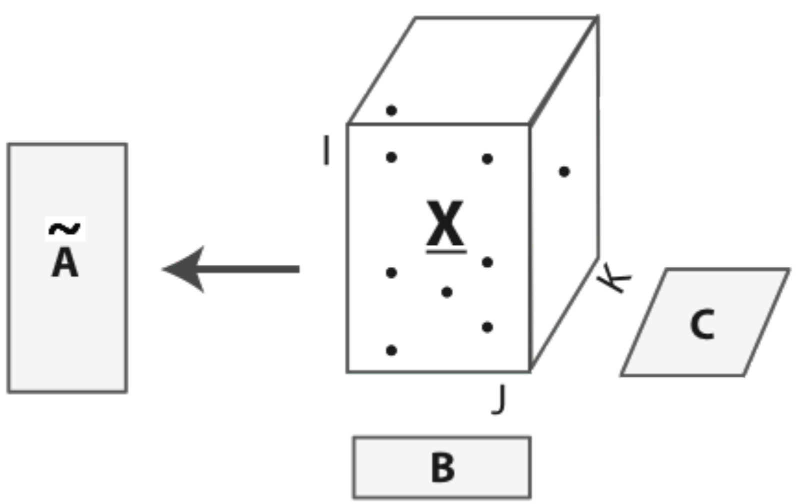
MTTKRP Operation

- Matriced Tensor Times Khatri-Rao Product (MTTKRP)

Tensor Matricization   Khatri-Rao Product

$$\tilde{\mathbf{A}} \leftarrow \mathbf{X}_{(1)} \, (\mathbf{C} \odot \mathbf{B})$$

Updated Factor Matrix   Factor Matrix



MTTKRP Operation

- Matriced Tensor Times Khatri-Rao Product (MTTKRP)

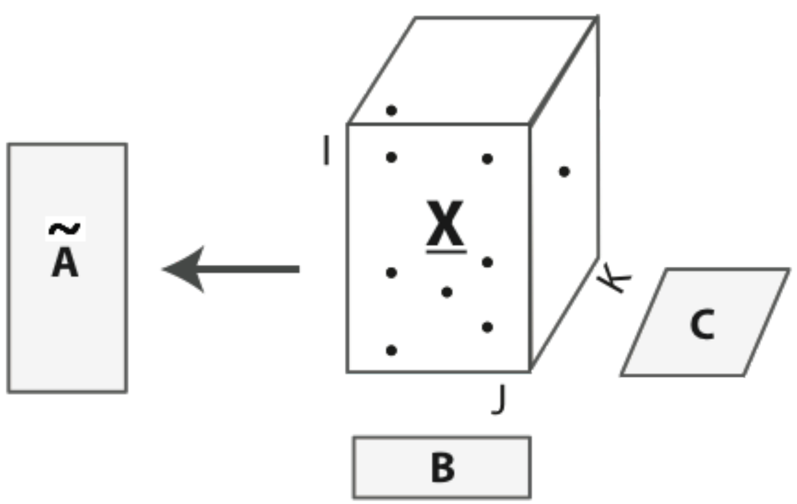
Tensor Matricization

Khatri-Rao Product

$$\tilde{\mathbf{A}} \leftarrow \mathbf{X}_{(1)} (\mathbf{C} \odot \mathbf{B})$$

Updated Factor Matrix

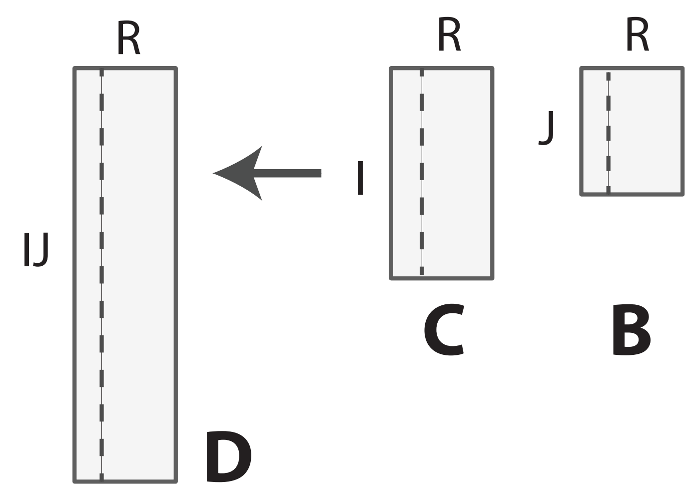
Factor Matrix



- Khatri-Rao Product

$$\mathbf{D} \leftarrow \mathbf{C} \odot \mathbf{B}$$

Matrix



CP-APR

for $k = 1, 2, \dots, k_{\max}$ **do**
  isConverged $\leftarrow$ true
  for $n = 1, \dots, N$ **do**

$$\mathbf{S}(i,r) \leftarrow \begin{cases} \kappa, & \text{if } k > 1, \mathbf{A}^{(n)}(i,r) < \kappa_{\text{tol}}, \text{ and } \mathbf{\Phi}^{(n)}(i,r) > 1, \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbf{B} \leftarrow (\mathbf{A}^{(n)} + \mathbf{S})\mathbf{\Lambda}$$

$$\mathbf{\Pi} \leftarrow \left(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)}\right)^{\mathsf{T}}$$

    for $\ell = 1, 2, \dots, \ell_{\max}$ **do** ▷ subproblem loop

$$\mathbf{\Phi}^{(n)} \leftarrow \left(\mathbf{X}_{(n)} \oslash \left(\max(\mathbf{B\Pi}, \epsilon)\right)\right) \mathbf{\Pi}^{\mathsf{T}}$$

      if $|\min(\mathbf{B}, \mathbf{E} - \mathbf{\Phi}^{(n)})| < \tau$ **then**
        break
      end if
      isConverged $\leftarrow$ false
      $\mathbf{B} \leftarrow \mathbf{B} * \mathbf{\Phi}^{(n)}$
    end for
    $\boldsymbol{\lambda} \leftarrow \mathbf{e}^{\mathsf{T}}\mathbf{B}$
    $\mathbf{A}^{(n)} \leftarrow \mathbf{B\Lambda}^{-1}$
  end for
  if isConverged = true **then**
    break
  end if
end for

Bottleneck:
element-wise product

Bottleneck of CP-APR

- Consider CP-APR MU

$$\mathbf{\Phi}^{(n)} \leftarrow \left(\mathbf{X}_{(n)} \oslash \left(\max(\mathbf{B}\mathbf{\Pi}, \epsilon)\right)\right) \mathbf{\Pi}^{\mathsf{T}}$$

$$\mathbf{\Pi} \leftarrow \left(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)}\right)^{\mathsf{T}}$$

Stored in sparse format, size nnz * R

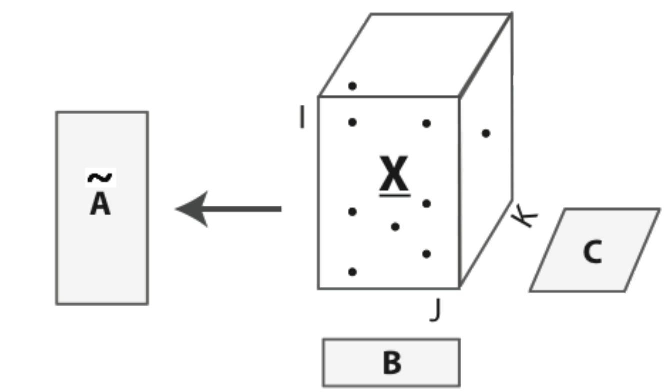
Outline

- Background
- HiCOO Format
- Multicore CP-ALS
- Distributed CPDs

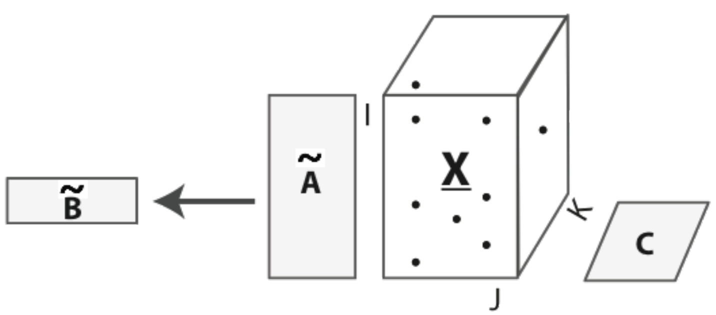
Mode Orientation

Tensor decomposition

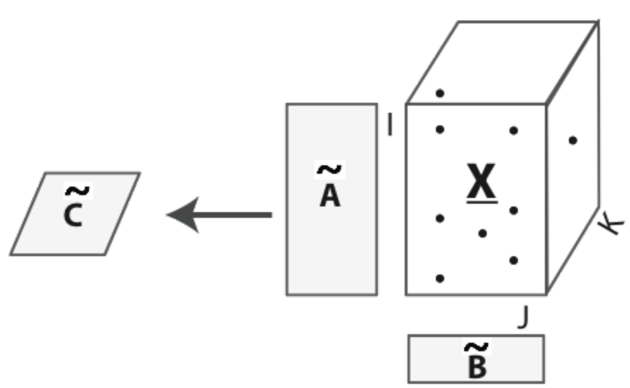
<u>Kernel in Mode-1</u>



<u>Kernel in Mode-2</u>



<u>Kernel in Mode-3</u>



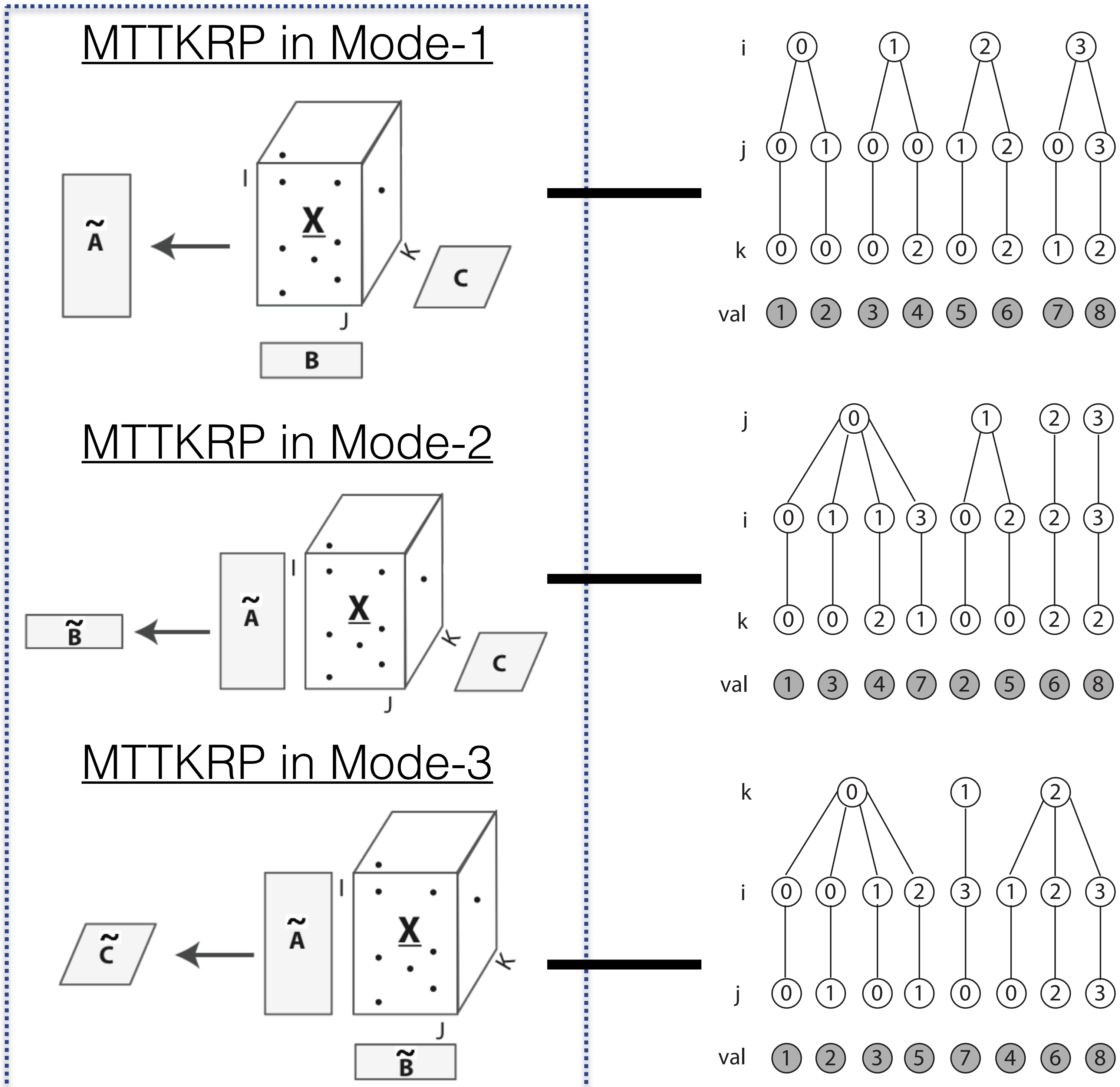
	Mode Oriented	Neutral Mode Orientation
	Mode-1 oriented (CSF)	Coordinate (COO)
Kernel in Mode-1	**Efficient**	**Inefficient**
Kernel in Mode-2	**?**	**?**
Kernel in Mode-3	**?**	**?**

Mode Oriented Formats

- Three CSF/F-COO representations are required/preferred for the three MTTKRPs.

CP Decomposition

More storage



CSF-1

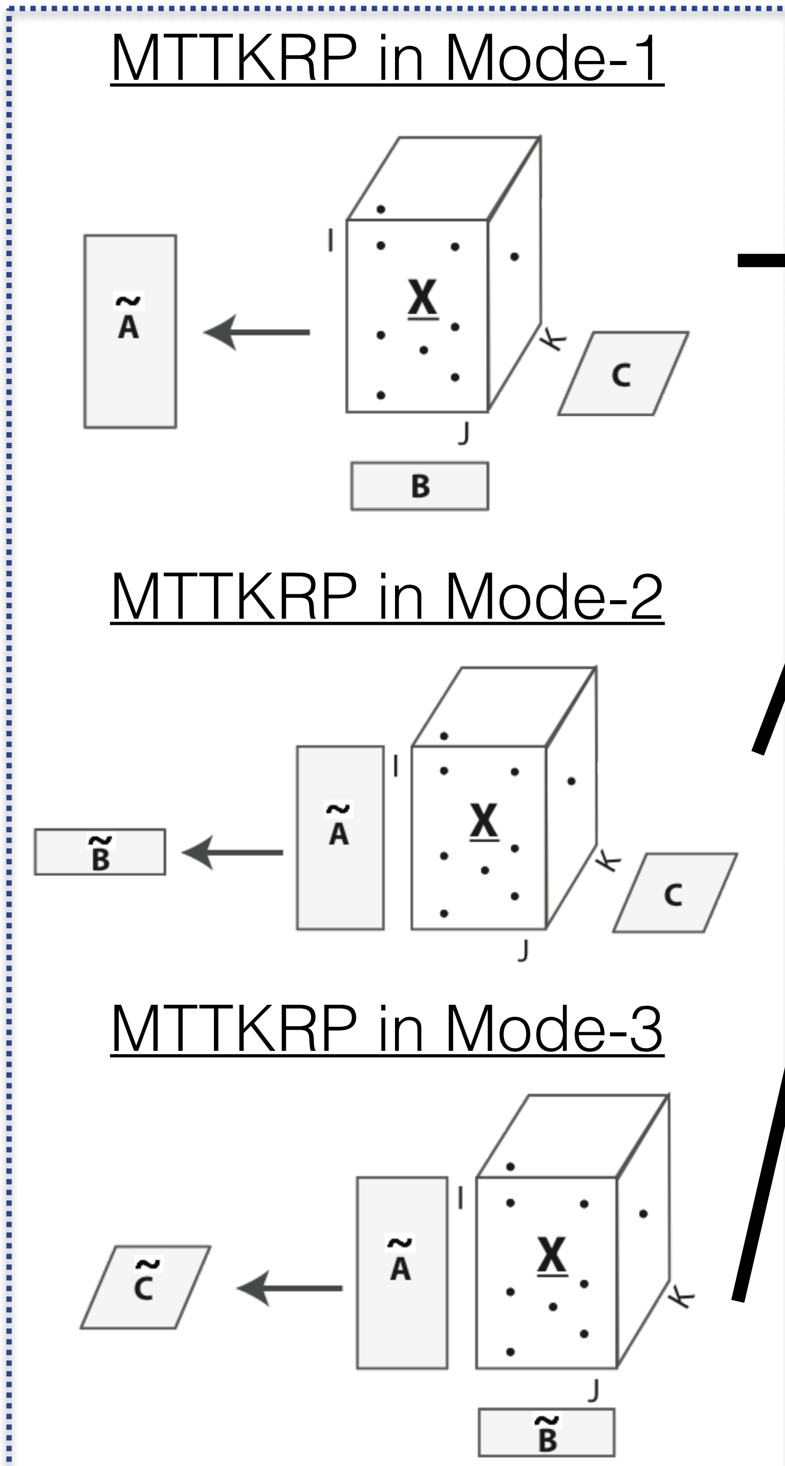
CSF-2

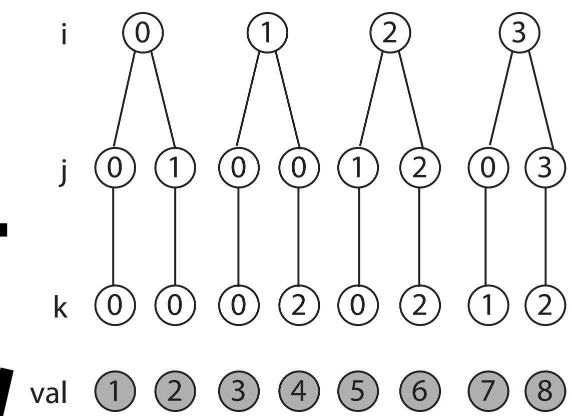
CSF-3

Mode Oriented Formats

- Three CSF/F-COO representations are required/preferred for the three MTTKRPs.

CP Decomposition



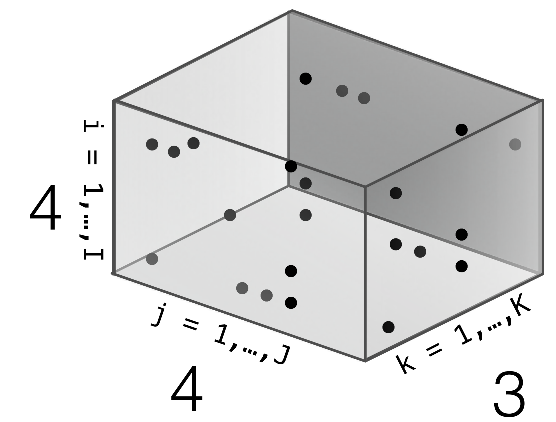


CSF-1

Performance payoff

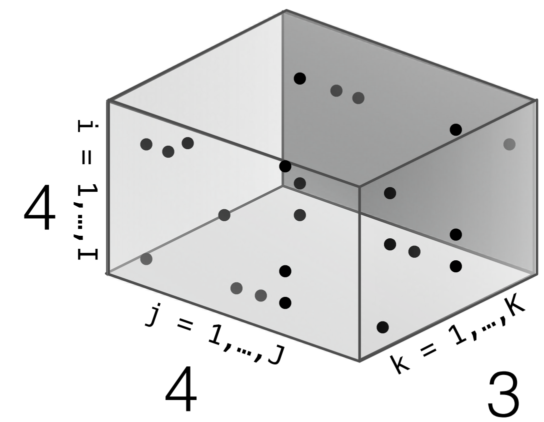
Current Sparse Tensor Formats

- **COO**: coordinate formats [Bader et al., 2006]
- **CSF**: Compressed Sparse Fibers, extension of CSR. [Smith et al. 2015]
- **F-COO**: Flagged COO format [Liu et al., 2017]



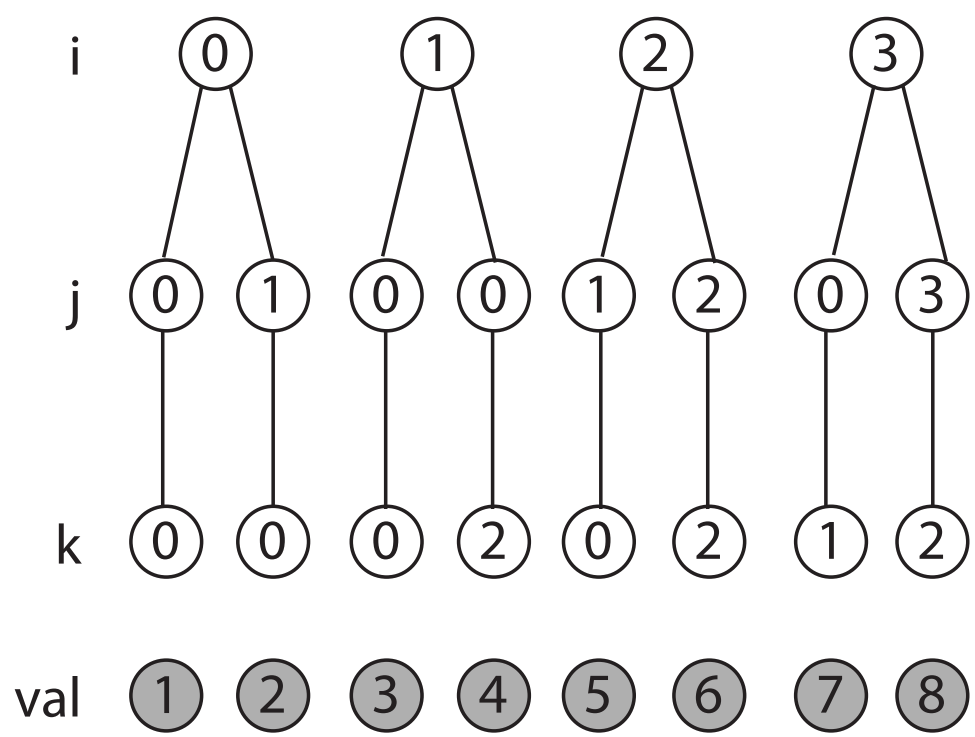
Current Sparse Tensor Formats

- **COO**: coordinate formats [Bader et al., 2006]
- **CSF**: Compressed Sparse Fibers, extension of CSR. [Smith et al. 2015]
- **F-COO**: Flagged COO format [Liu et al., 2017]



i	j	k	val
0	0	0	1
0	1	0	2
1	0	0	3
1	0	2	4
2	1	0	5
2	2	2	6
3	0	1	7
3	3	2	8

(a) COO



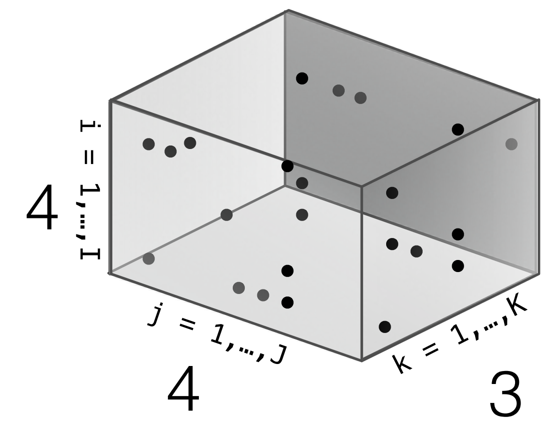
(b) CSF

	bf	j	k	val
sf[0]=1	1	0	0	1
	0	1	0	2
	1	0	0	3
	0	0	2	4
sf[1]=1	1	1	0	5
	0	2	2	6
	1	0	1	7
	0	3	2	8

(c) F-COO

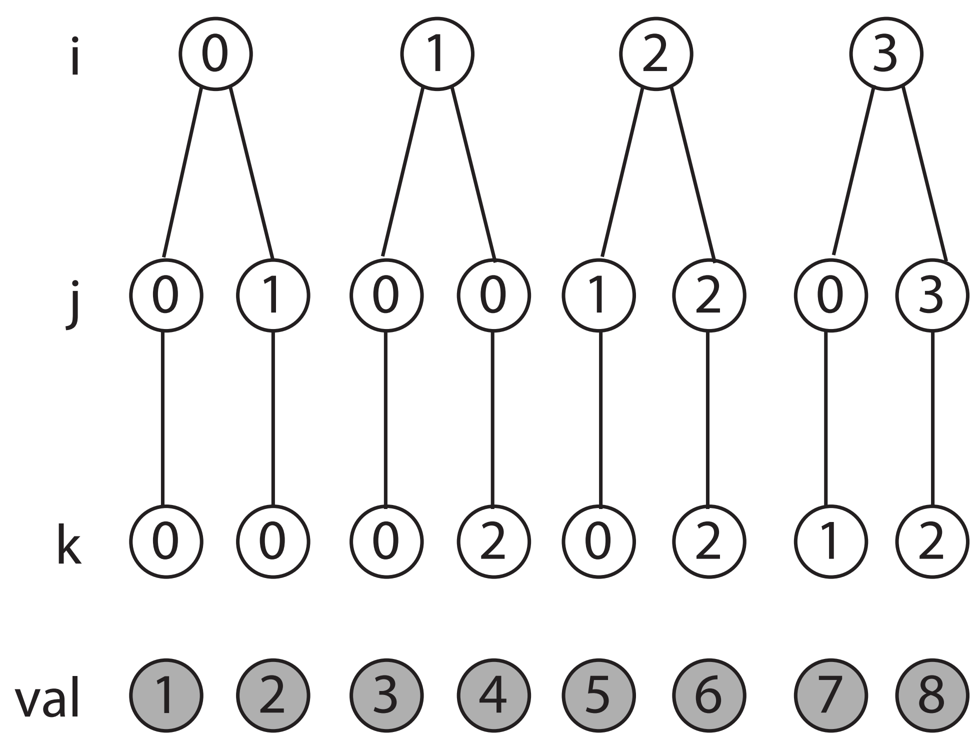
Current Sparse Tensor Formats

- **COO**: coordinate formats [Bader et al., 2006]
- **CSF**: Compressed Sparse Fibers, extension of CSR. [Smith et al. 2015]
- **F-COO**: Flagged COO format [Liu et al., 2017]



i	j	k	val
0	0	0	1
0	1	0	2
1	0	0	3
1	0	2	4
2	1	0	5
2	2	2	6
3	0	1	7
3	3	2	8

(a) COO



(b) CSF

	bf	j	k	val
sf[0]=1	1	0	0	1
	0	1	0	2
	1	0	0	3
	0	0	2	4
sf[1]=1	1	1	0	5
	0	2	2	6
	1	0	1	7
	0	3	2	8

(c) F-COO

neutral
mode orientation

mode oriented,
prefer different representations for different modes.

HiCOO Format

- Store a sparse tensor in units of small sparse blocks, saving storage
  - Shorten the bit-length of element indices
  - Compress the number of block indices
  - an improvement of the Compressed Sparse Blocks (CSB) format

32-bit

i	j	k	val
0	0	0	1
0	1	0	2
1	0	0	3
1	0	2	4
2	1	0	5
2	2	2	6
3	0	1	7
3	3	2	8

(a) COO

32-bit **8-bit**

	bptr	bi	bj	bk	ei	ej	ek	val
B0	0	0	0	0	0	0	0	1
					0	1	0	2
					1	0	0	3
B1	3	0	0	1	1	0	0	4
B2	4	1	0	0	0	1	0	5
					1	0	1	7
B3	6	1	1	1	0	0	0	6
					1	1	0	8

(b) HiCOO

$i = bi * B + ei$

<u>For the tensor</u>: Reduce its storage and memory footprints

<u>For matrices</u>: Better data locality

Construct HiCOO Representation

- Morton-order sorting for the input COO tensor.
- Partition to blocks and determine each block location.
- Construct HiCOO representation by compressing index length.

bits: β_{int}

i	j	k	val
0	0	0	1
0	1	0	2
1	0	0	3
1	0	2	4
2	1	0	5
2	2	2	6
3	0	1	7
3	3	2	8

COO

Z-Order Sorting →

β_{int}

i	j	k	val
0	0	0	1
0	1	0	2
1	0	0	3
1	0	2	4
2	1	0	5
3	0	1	7
2	2	2	6
3	3	2	8

Partition →

β_{long} β_{int}

bptr	i	j	k	val
0	0	0	0	1
	0	1	0	2
	1	0	0	3
3	1	0	2	4
4	2	1	0	5
	3	0	1	7
6	2	2	2	6
	3	3	2	8

Compress →

β_{long} β_{int} β_{byte}

	bptr	bi	bj	bk	ei	ej	ek	val
B0	0	0	0	0	0	0	0	1
					0	1	0	2
					1	0	0	3
B1	3	0	0	1	1	0	0	4
B2	4	1	0	0	0	1	0	5
					1	0	1	7
B3	6	1	1	1	0	0	0	6
					1	1	0	8

HiCOO

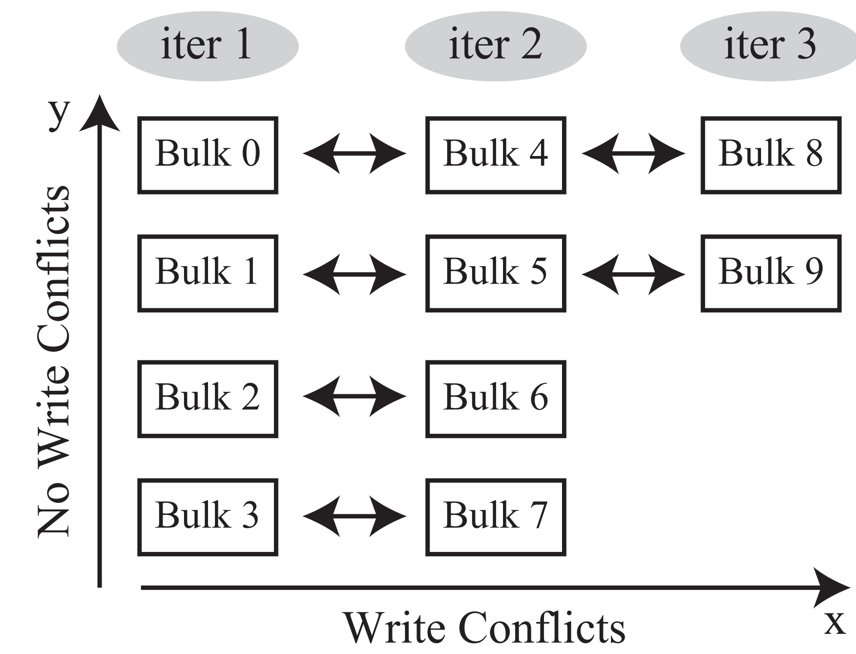
$i = bi*B+ei;\ \ j = bj*B+ej;\ \ k = bk*B+ek$

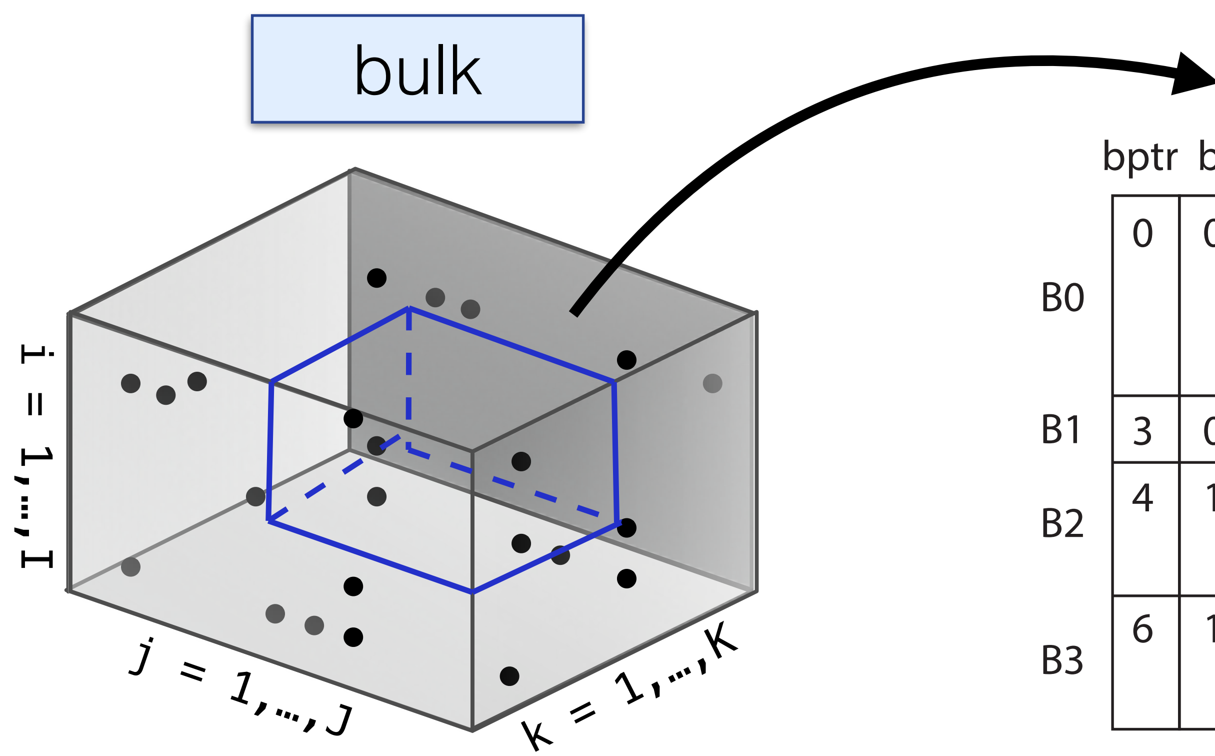
Outline

- Background
- HiCOO Format
- Multicore CP-ALS
- Distributed CPDs

Two-level Blocking for Efficient Thread Parallelism

- Use two-level blocking strategy
  - Large bulks (logical) + small blocks (physical)
  - To avoid using locks, we add a bit extra storage for scheduling information.





	bptr	bi	bj	bk	ei	ej	ek	val
B0	0	0	0	0	0	0	0	1
					0	1	0	2
					1	0	0	3
B1	3	0	0	1	1	0	0	4
B2	4	1	0	0	0	1	0	5
					1	0	1	7
B3	6	1	1	1	0	0	0	6
					1	1	0	8

HiCOO

Platform and Dataset

- **Platform**: Intel Xeon CPU E7-4850 platform consisting 56 physical cores with gcc 5.4.1 and parallelized by OpenMP.
- **Dataset**: FROSTT [Smith et al. 2017] and HaTen2 [Jeon et al. 2015]

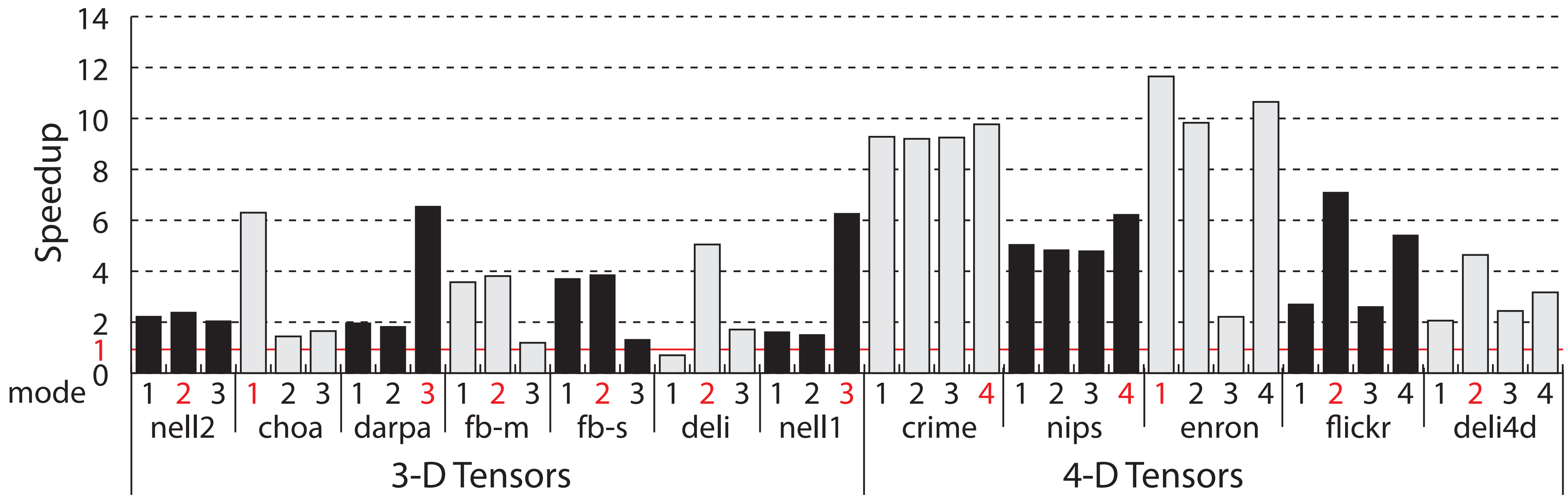
DESCRIPTION OF SPARSE TENSORS.

Tensors	Order	Dimensions	#Nonzeros	Density
nell2	3	$12K \times 9K \times 29K$	77M	2.4×10^{-5}
choa	3	$712K \times 10K \times 767$	27M	5.0×10^{-6}
darpa	3	$22K \times 22K \times 24M$	28M	2.4×10^{-9}
fb-m	3	$23M \times 23M \times 166$	100M	1.1×10^{-9}
fb-s	3	$39M \times 39M \times 532$	140M	1.7×10^{-10}
deli	3	$533K \times 17M \times 2.5M$	140M	6.1×10^{-12}
nell1	3	$3M \times 2M \times 25M$	144M	9.1×10^{-13}
crime	4	$6K \times 24 \times 77 \times 32$	5M	1.5×10^{-2}
nips	4	$2K \times 3K \times 14K \times 17$	3M	1.8×10^{-6}
enron	4	$6K \times 6K \times 244K \times 1K$	54M	5.5×10^{-9}
flickr	4	$320K \times 28M \times 2M \times 731$	113M	1.1×10^{-14}
deli4d	4	$533K \times 17M \times 2M \times 1K$	140M	4.3×10^{-15}

J. Li, J. Sun, R. Vuduc. “HiCOO: Hierarchical Storage of Sparse Tensors”. 2018

Multicore MTTKRP

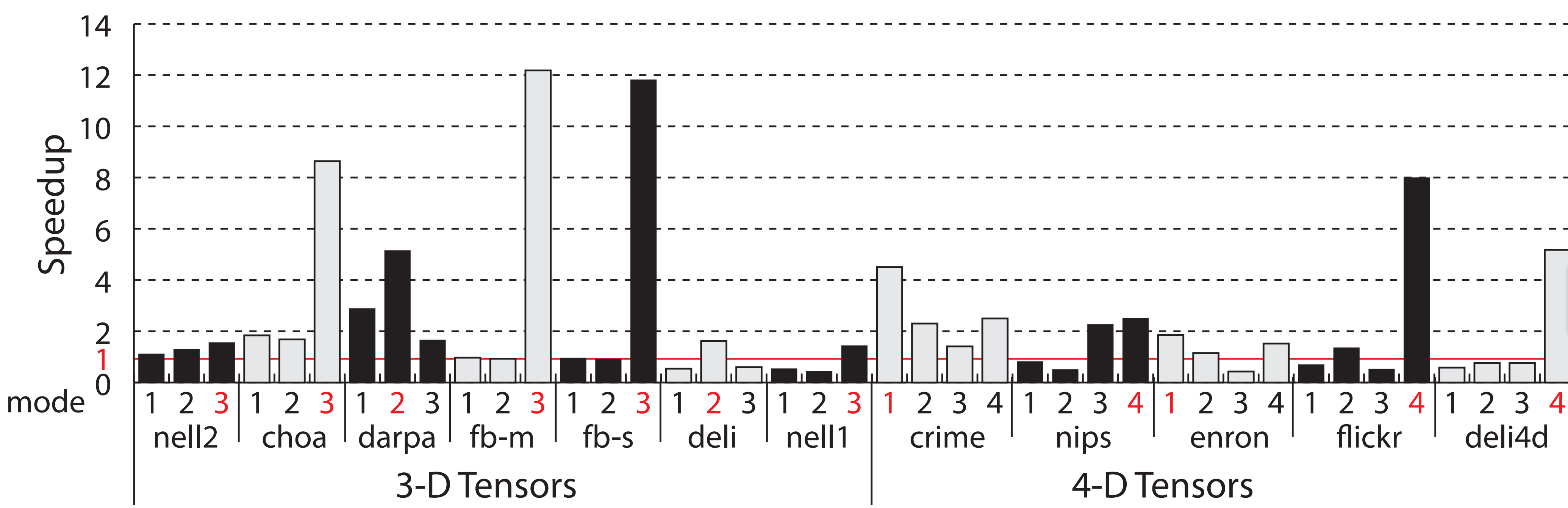
- ParTI! library: Speedups of HiCOO over COO



0.8x

- SPLATT library: Speedups of HiCOO over CSF

Storage size:

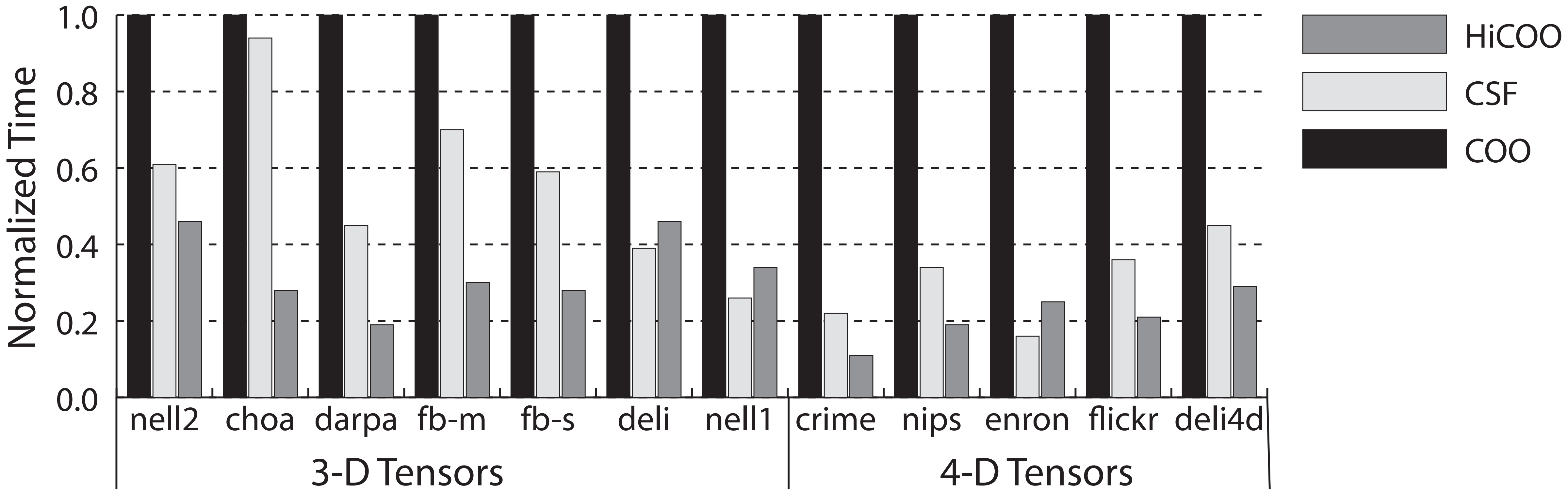


0.3x

comparable

Multicore CP-ALS

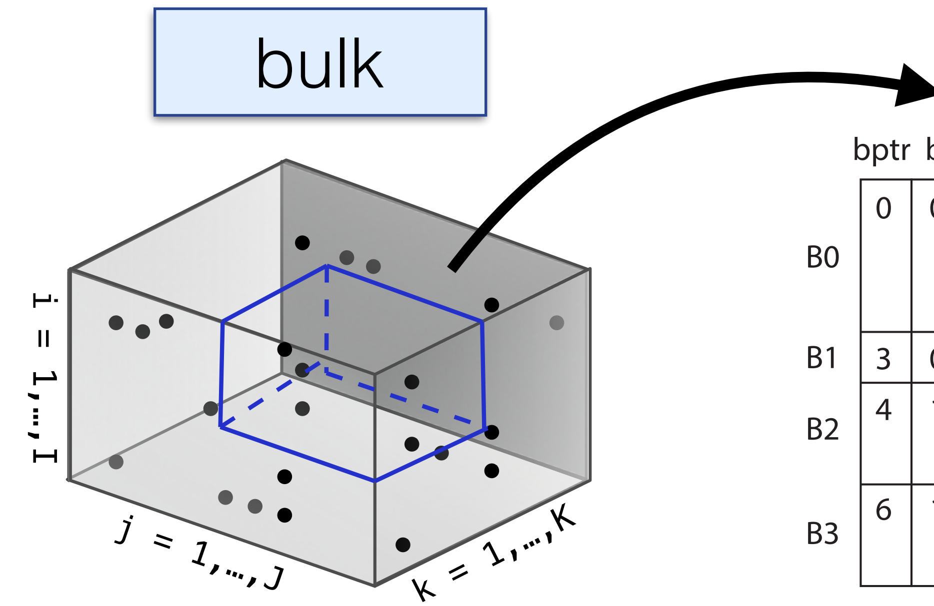
- HiCOO outperforms COO by up to 9.4× and CSF up to 3.4×.



Outline

- Background
- HiCOO Format
- Multicore CP-ALS
- Distributed CPDs

Data Distribution



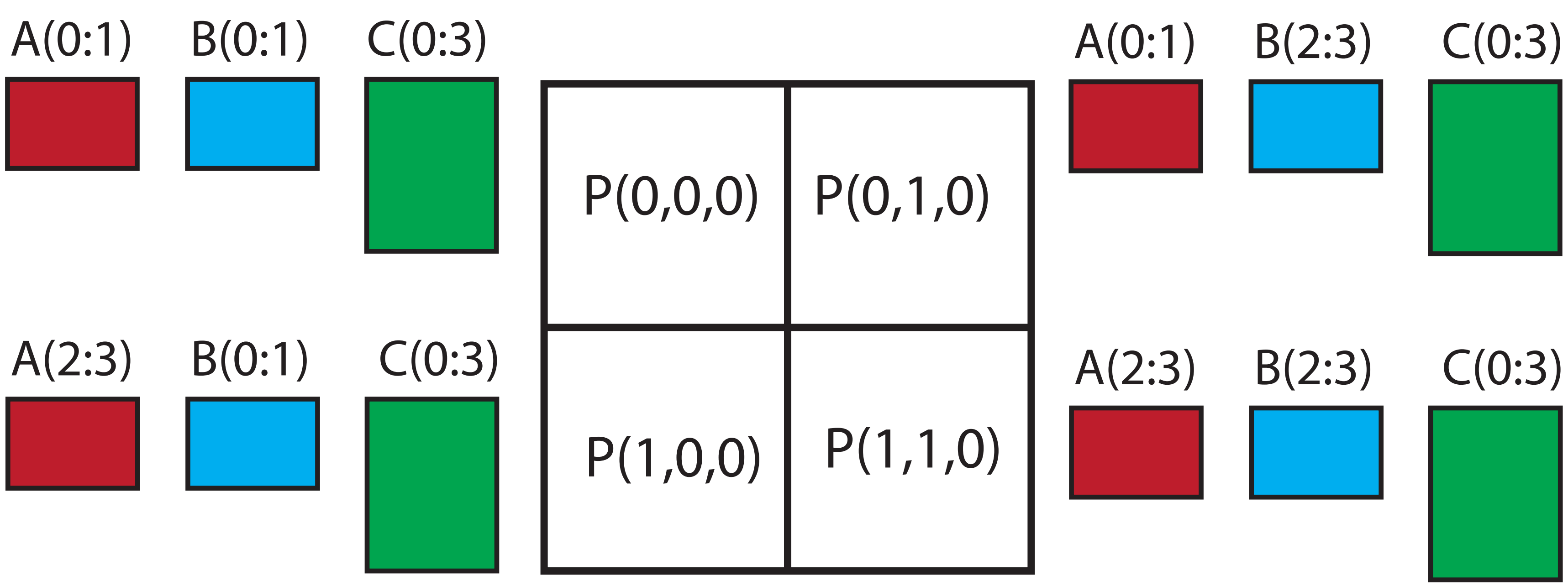
	bptr	bi	bj	bk	ei	ej	ek	val
B0	0	0	0	0	0	0	0	1
					0	1	0	2
					1	0	0	3
B1	3	0	0	1	1	0	0	4
B2	4	1	0	0	0	1	0	5
					1	0	1	7
B3	6	1	1	1	0	0	0	6
					1	1	0	8

HiCOO

- Tensor:
  - Two-level blocking
  - Bulk level: node partitioning to ensure nonzero balance
  - Block level: ensure fast local computation
- Factor matrices
  - Each processor owns the portion of factor matrices its local tensor needs.
  - Some duplication exists.

Data Distribution Cont.

- X: 4*4*4 sparse tensor
- Processors are split to 2*2*2 grids.
- Use sub-communicator to update factor matrices.



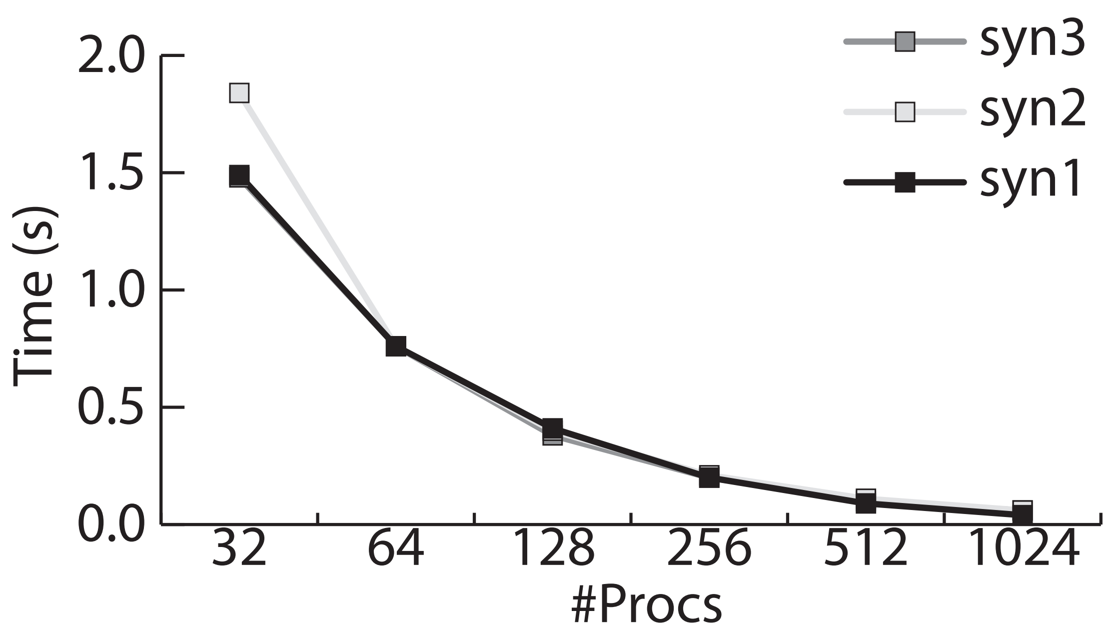
Platform and Dataset

- **Platform**: Cori supercomputer in NERSC, which is Cray XC40. Each node is Intel Xeon CPU E5-2698 v3 ("Haswell") consisting 32 physical cores. We use Intel MPI 2018 version.
  - We test 32 to 1024 processors with 32 processors per node.
- **Dataset**: Synthetic tensors from Poisson distribution.

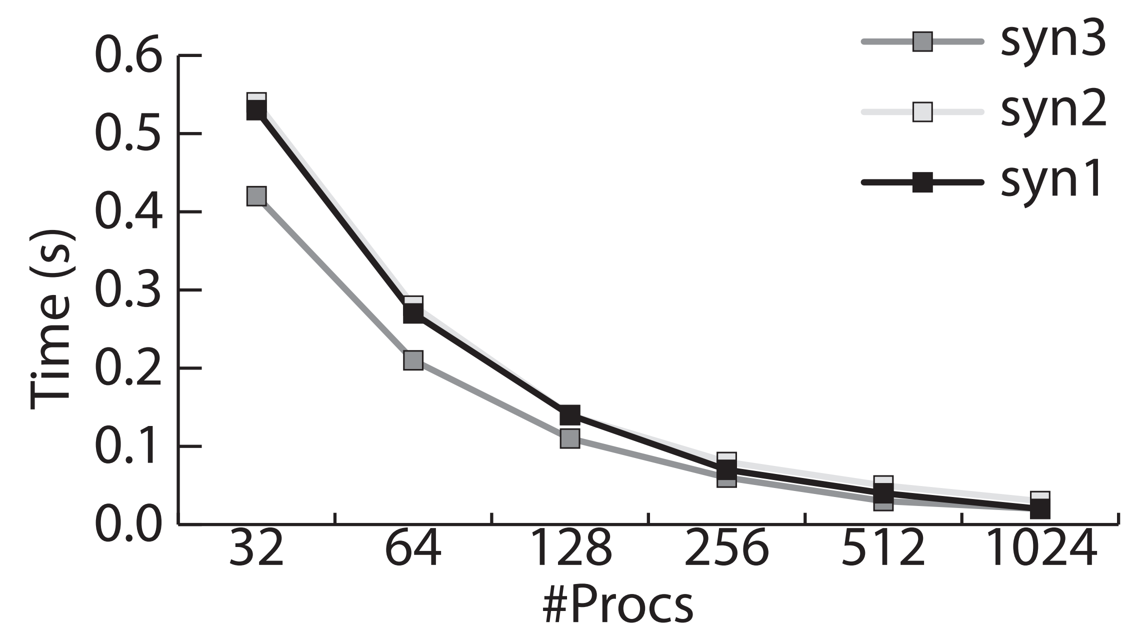
Tensors	Order	Dimensions	#Nonzeros	Density
syn1	3	$10K \times 10K \times 10K$	100M	1.0×10^{-4}
syn2	3	$50K \times 50K \times 50K$	125M	1.0×10^{-6}
syn3	3	$40K \times 40K \times 40K$	128M	2.0×10^{-6}

Distributed MTTKRP Performance

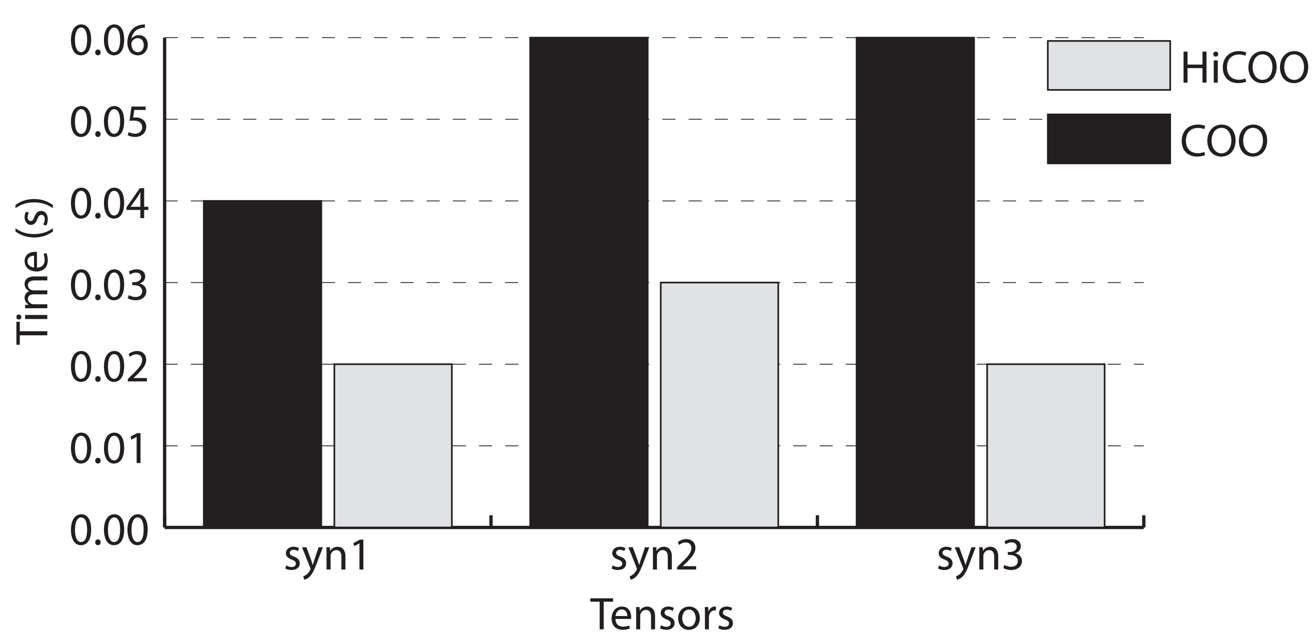
COO



HiCOO

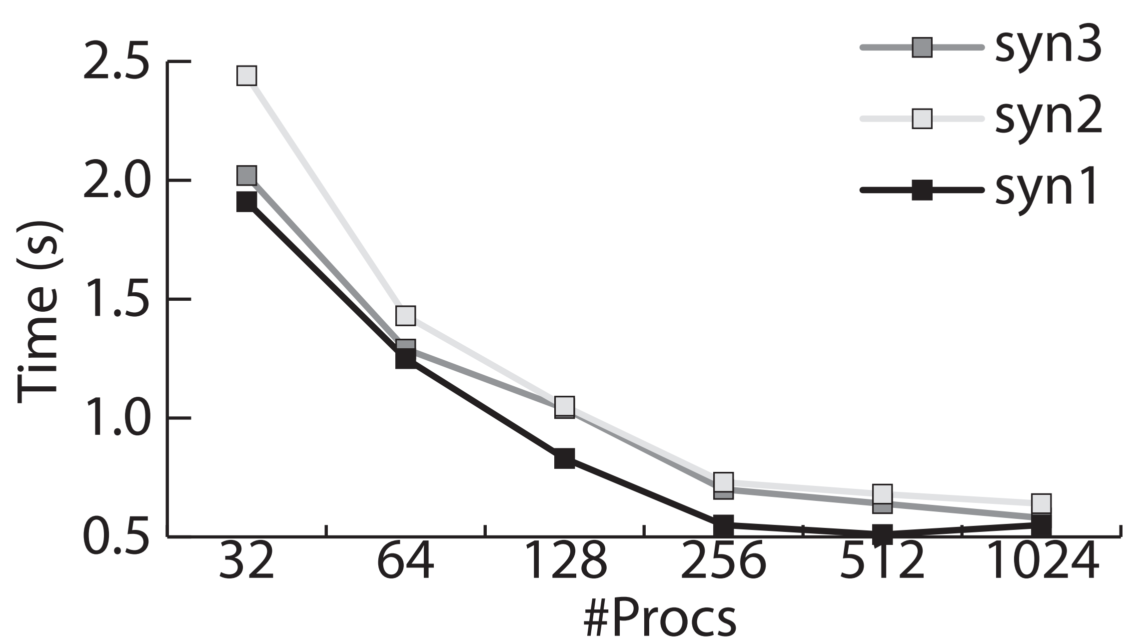


MTTKRP time using 1024 procs

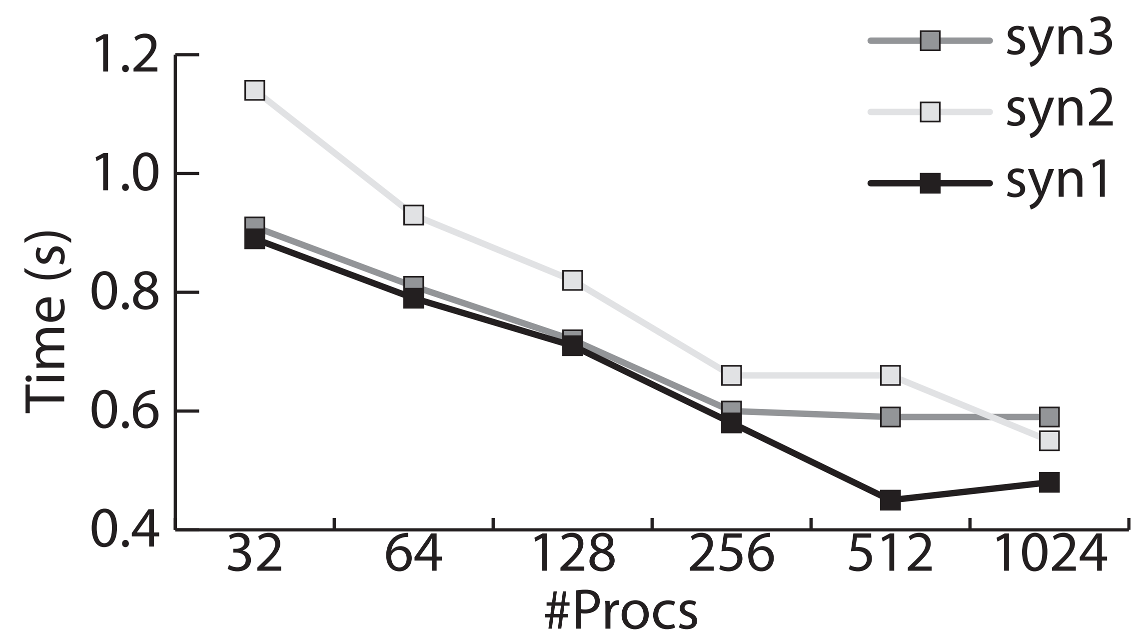


Distributed CP-ALS Performance

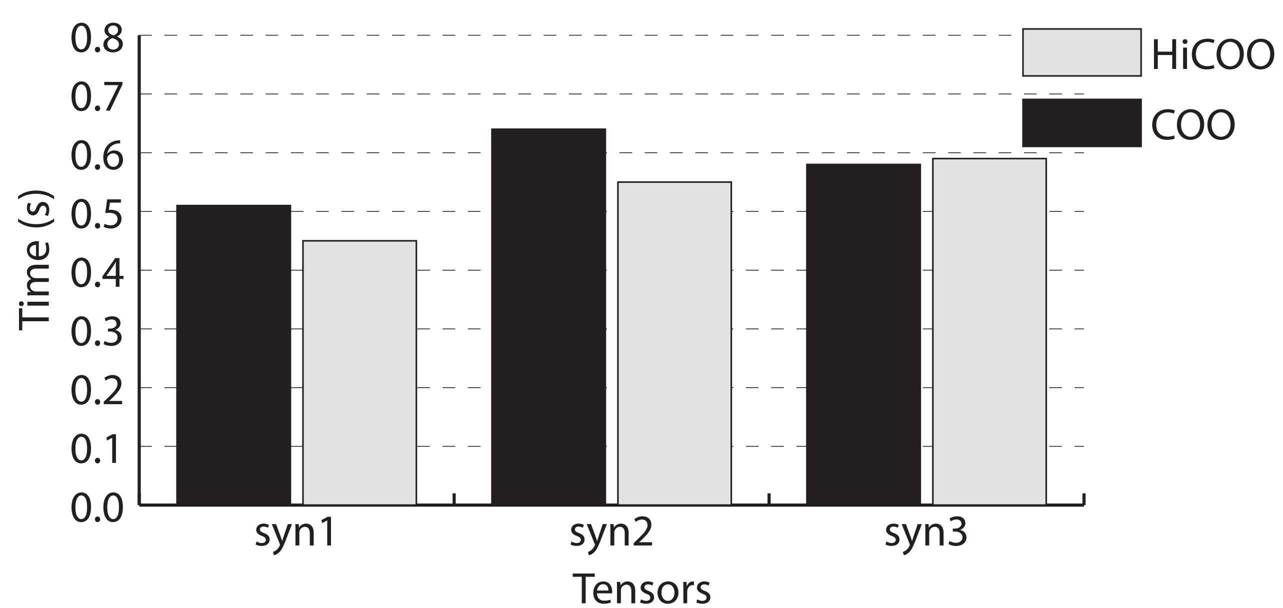
COO



HiCOO

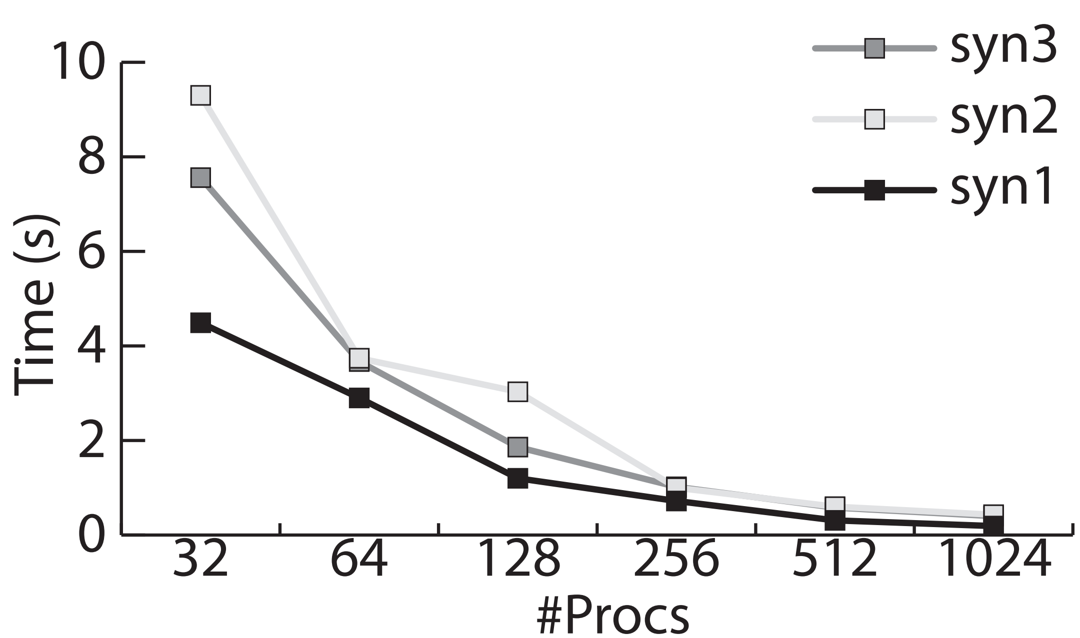


CP-ALS time using 1024 procs

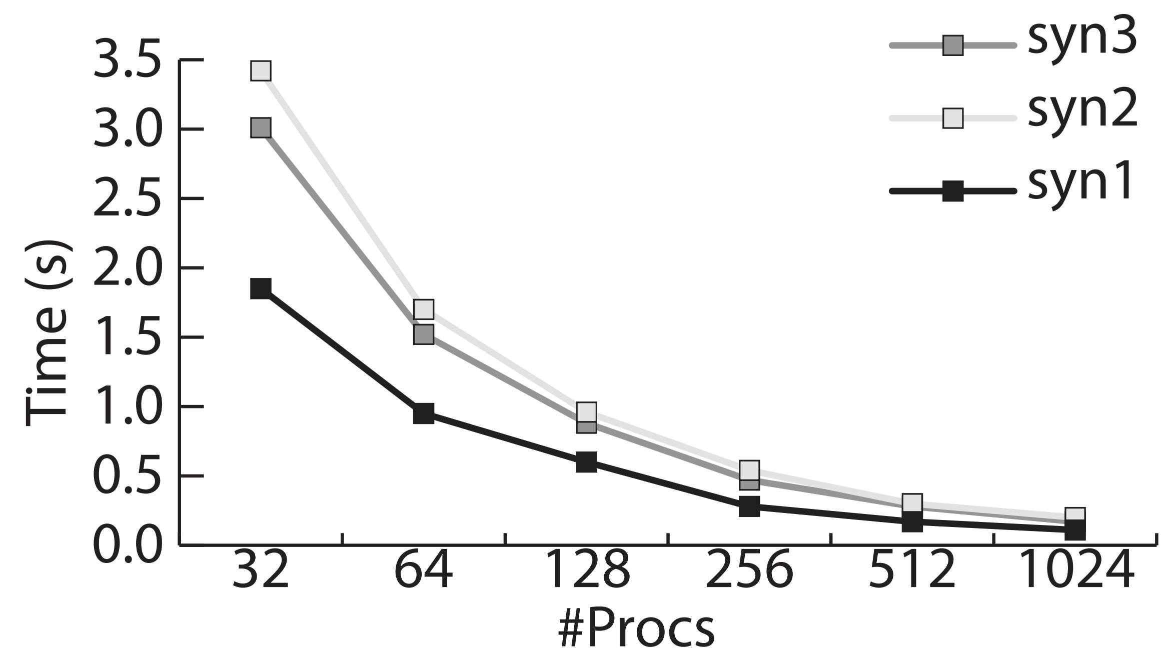


Distributed CompPhi Performance

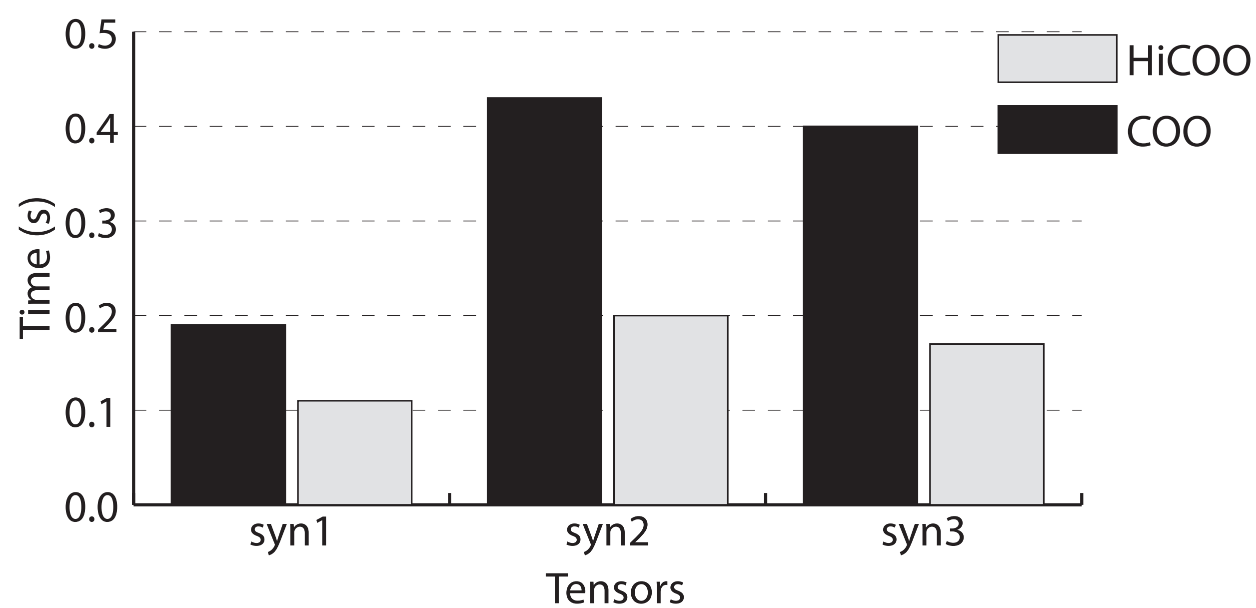
COO



HiCOO

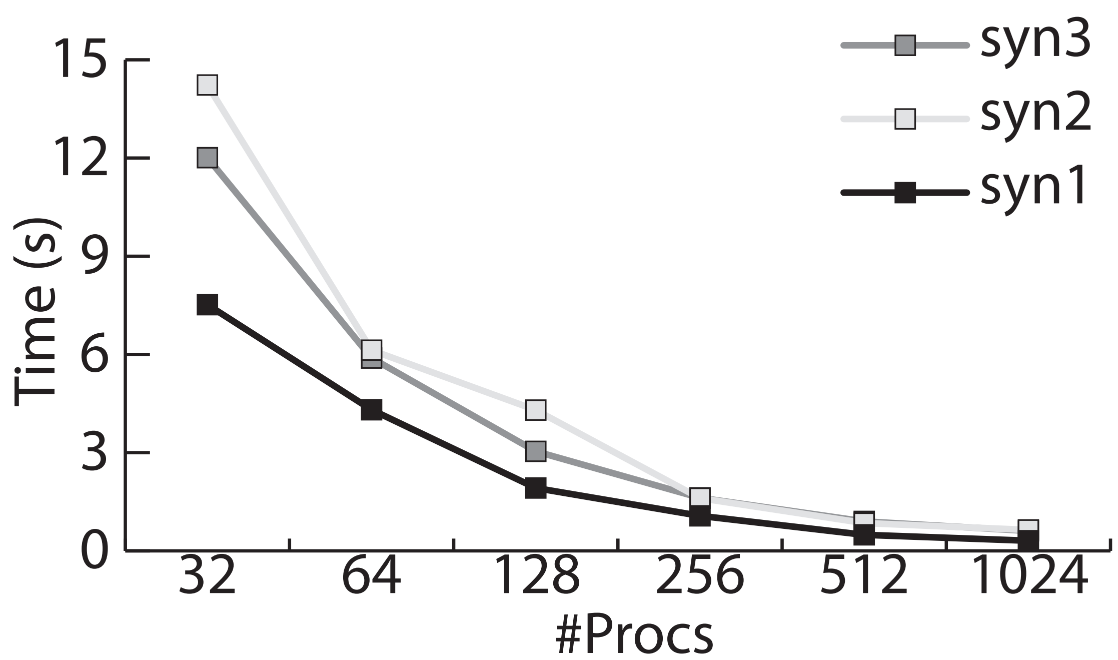


CompPhi time using 1024 procs

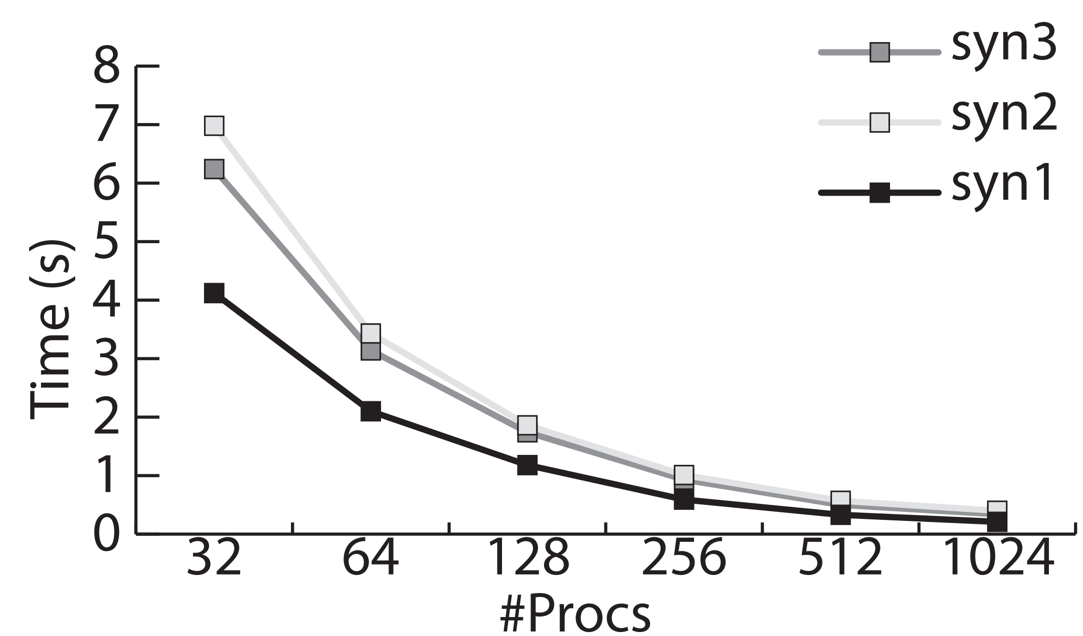


Distributed CP-APR Performance

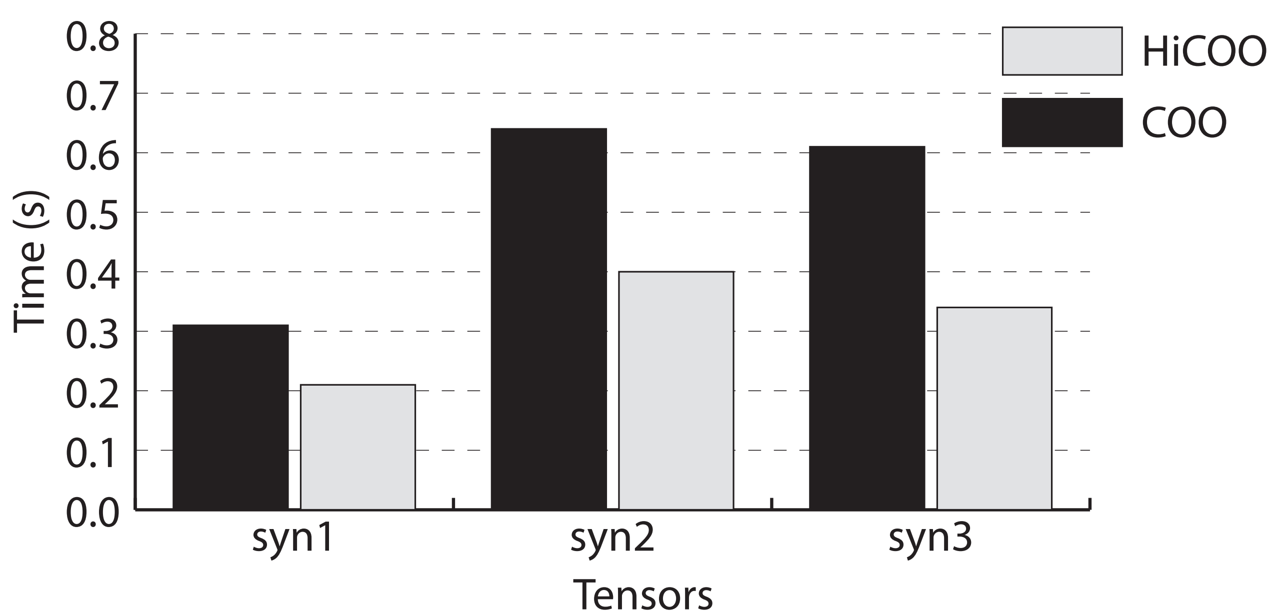
COO



HiCOO



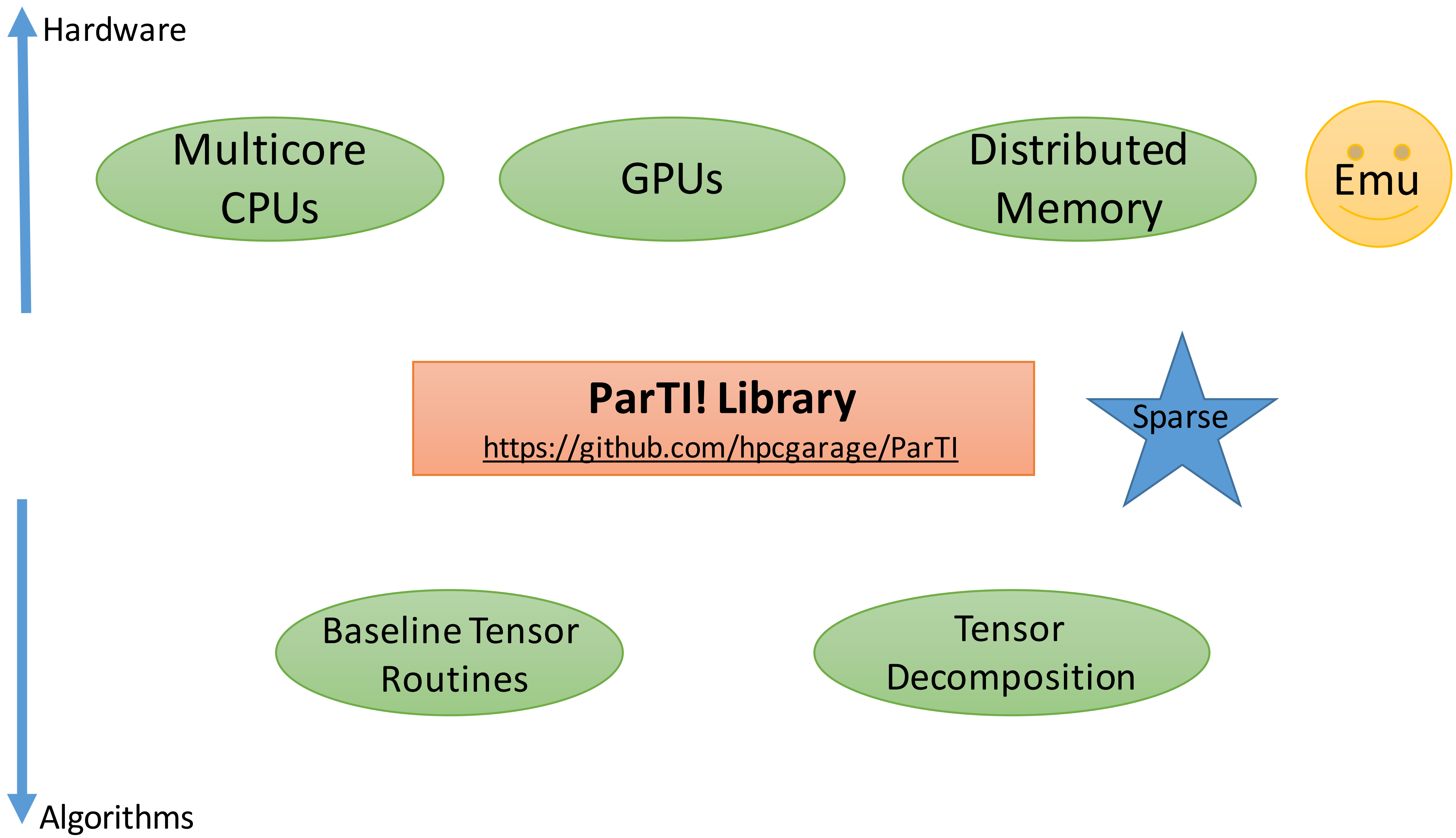
CP-APR time using 1024 procs



Conclusion and Future Work

- Conclusion
  - **HiCOO WORKS!**
- Future work
  - Combine multicore and distributed parallelism together to build hybrid MPI+OpenMP HiCOO CPDs.
  - Also include our GPU implementations.
  - Consider CP-APR algorithms to Newton-based optimization methods.

ParTI! Project



Code: https://github.com/hpcgarage/ParTI

Email: jiajiali@gatech.edu

Contributors

Jiajia Li

PhD, GaTech & PNNL*

* From Aug 2018

Yuchen Ma

Undergrad, HDU

Nick Liu

Undergrad, GaTech

Srinivas Eswar

PhD, GaTech

Junghyun Kim

Undergrad, GaTech

Richard Vuduc

Professor, GaTech

Acknowledgement

SunLab
Enlighten the World

NSF

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

Sandia National Laboratories

Emu TECHNOLOGY

Related Work

- S. Smith et al. “A Medium-Grained Algorithm for Distributed Sparse Tensor Factorization”, IPDPS’16
- O. Kaya et al. “Scalable Sparse Tensor Decompositions in Distributed Memory Systems”, SC’15
- E. Solomonik et al. “Sparse Tensor Algebra as a Parallel Programming Model”, TR, 2015
- S. Rajbhandari et al. “A communication-optimal framework for contracting distributed tensors”, SC'14
- W. Austin et al. “Parallel Tensor Compression for Large-Scale Scientific Data”, IPDPS’16
- J. Choi et al. “DFacTo: Distributed Factorization of Tensors”, NIPS’14