Efficient Large-Scale Sparse Multilinear Algebra Computations

Somesh Singh

Post-doctoral Researcher, ROMA team, LIP, ENS de Lyon

Topal team candidate

High Performance Computing (HPC), Tensor Computations, AI

April 09, 2024

Professional History

Legend:

- Tensor computation
- Graph processing
- Parallelization
- GPU
- Application

Google Summer of Code participant | May 2018 – August 2018

European Organization for Nuclear Research (CERN)

- Parallelizing **SixTrackLib**, a particle-tracking library used for the Large Hedron Collider (LHC)

[ICAP 2018, IJMPA 2019]

Research intern | September 2020 – December 2020

Microsoft Research (MSR) India

- Approximate Nearest Neighbor (ANN) search on a graph index using a GPU

[Manuscript 2024 (under review), Patent application in progress]

PhD and Master's | July 2014 – June 2021

Indian Institute of Technology Madras, India

- Parallel graph processing on graphics processing unit (GPU)
- Inexact computations on graphs
- Performance-accuracy trade-off

[TMSCS 2018, PPoPP 2019, ICPP 2020, TODAES 2021, GECCO 2023]

Post-doctoral researcher | September 2021 – Present

ROMA team at LIP, ENS Lyon

- High-performance parallel sparse tensor (and matrix) computations
- Efficient hashing-based methods for sparse tensor computations

[GrAPL 2022 (IPDPSW), JEA 2022, ESA 2023]

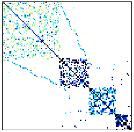
Research Context and Motivation

Computations on Tensors

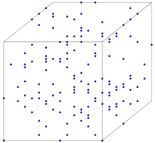
Computations on Sparse Tensors

- Tensors are multi-dimensional arrays and model multi-dimensional data

1D Tensor / Vector



2D Tensor / Matrix



3D Tensor / Cube

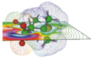
- Sparsity enables scalability

- Tensors vary in dimensionality, sparsity and size

Internet & Social media

Recommendation systems

Circuit Simulation



Computational Chemistry

Problems in Statistics

Deep Learning

Density: (log scale) >10^{-6}% 10^{-4}% 10^{-2}% 1% 10% 50% 100%

- A single data structure and algorithm is inadequate for full spectrum of sparsity!

Graphics Source: Google Images

Past Research Work

PhD research

- Parallel graph processing on GPU: TMSCS 2018, PPoPP 2019, ICPP 2020, TODAES 2021, GECCO 2023
- Performance-accuracy tradeoff

Interaction with Industry

- Research internship at Microsoft Research (MSR) India
  - Approximate Nearest Neighbor (ANN) search on a graph index using a GPU: manuscript (arxiv 2024) (under review), Patent application in progress
- Google Summer of Code participant with European Organization for Nuclear Research (CERN)
  - Parallelizing **SixTrackLib**, a particle-tracking library used for the Large Hedron Collider (LHC): ICAP 2018, IJMPA 2019

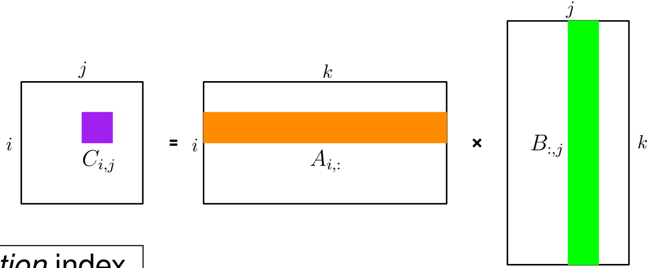
Post-doctoral research

- High-performance parallel sparse tensor computations: GrAPL 2022 (IPDPSW), JEA 2022, ESA 2023, manuscript ready for submission
- Efficient hashing-based methods for sparse tensor computations

Efficient Parallel Sparse Tensor Contraction

- Tensor contraction is a higher-dimensional analog of matrix-matrix multiplication
- Multiplication of two matrices: $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{K \times J}$

$$\mathbf{C}_{ij} = \sum_{k} \mathbf{A}_{ik} \cdot \mathbf{B}_{kj}$$

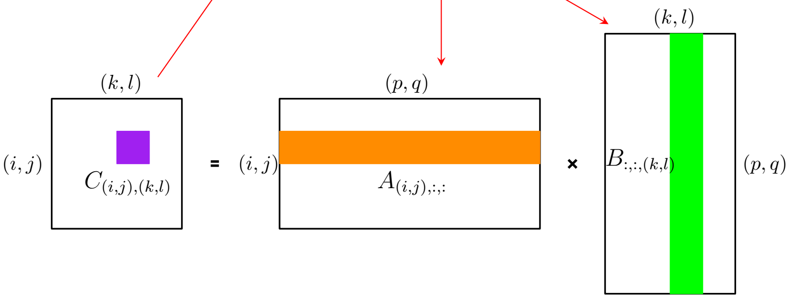


k is the *contraction* index

Sparse Tensor Contraction (SpGETT)

- Contraction of two tensors: $\mathcal{A} \in \mathbb{R}^{I \times J \times P \times Q}$ and $\mathcal{B} \in \mathbb{R}^{P \times Q \times K \times L}$ with p,q as contraction indices

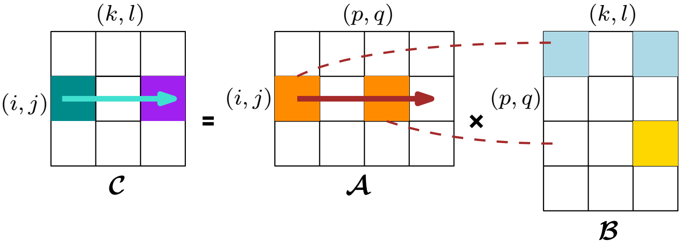
$$\mathcal{C}_{ijkl} = \sum_{p} \sum_{q} \mathcal{A}_{ijpq} \cdot \mathcal{B}_{pqkl}$$



Our formulation of SpGETT

- Contraction of two tensors: $\mathcal{A} \in \mathbb{R}^{I \times J \times P \times Q}$ and $\mathcal{B} \in \mathbb{R}^{P \times Q \times K \times L}$ with p,q as contraction indices

```
parfor nonempty subtensor 𝓐[i,j,:,:] do
    for nonzero 𝓐[i,j,p,q] in subtensor 𝓐[i,j,:,:] do
        for nonzero 𝓑[p,q,k,l] in subtensor 𝓑[p,q,:,:] do
            v ← 𝓐[i,j,p,q] · 𝓑[p,q,k,l]
            if 𝓒[i,j,:,:].lookup(⟨k,l⟩) = True then
                𝓒[i,j,k,l] ← 𝓒[i,j,k,l] + v
            else
                𝓒[i,j,:,:].insert(⟨k,l⟩)
                𝓒[i,j,k,l] ← v
```



$$\mathcal{C}_{ij::} = \sum_p \sum_q \mathcal{A}_{ijpq} \cdot \mathcal{B}_{pq::}$$

Our hashing method powering SpGETT

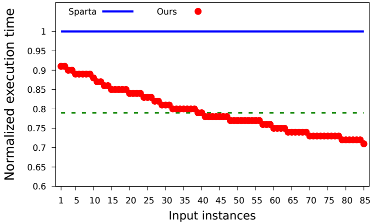
- Guaranteed $O(1)$ lookup per query in the worst-case
  - classical hashing methods guarantee average case $O(1)$ lookup
  - dense arrays guarantee $O(1)$ lookup, but are not space efficient
- Construction time is $O(n)$, in expectation
- Total storage space $O(n)$; less than $5n$ in practice
- Supports dynamic insertions, and multiple items may have identical hashing indices
- Our hashing method has utility beyond SpGETT
  - can be used for other computations on sparse tensors, and on dynamic graphs

J. Bertrand, F. Dufossé, **S. Singh** and B. Uçar, "Algorithms and Data Structures for Hyperedge Queries", ACM Journal of Experimental Algorithmics (JEA), vol. 27, no. 1, Article 1.13, 23 pages, 2022 [**ACM Results Reproduced Badge**]

* **Somesh Singh**, Bora Uçar, "Efficient Parallel Sparse Tensor Contraction" [Manuscript, code]

Experimental Evaluation

- Operation considered for evaluation: $\mathcal{C} = \mathcal{A}\mathcal{A}^T$
- Baseline: Sparta[§], the state-of-the-art for SpGETT
- Sequential execution: Ours* is always faster than Sparta, by 25% on average, on real-world tensors from FROSTT
- Parallel execution: Ours is always faster than Sparta, by 21% on average, on real-world tensors from FROSTT across thread counts of {16, 32, 48, 64, 80, 96}



threads = 96

* **Somesh Singh**, Bora Uçar, "Efficient Parallel Sparse Tensor Contraction" [Manuscript, code]

§ J. Liu, J. Ren, R. Gioiosa, D. Li, and J. Li, "Sparta: High-Performance, Element-Wise Sparse Tensor Contraction on Heterogeneous Memory", ACM PPoPP 2021.

Research Program

&

Integration to Topal

Research Program

Objective

Develop methodology and technology to enable large-scale data analytics in AI and scientific computing

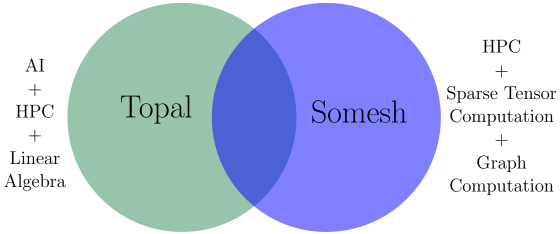
Approach

Design novel parallel algorithms and parallelization techniques for accelerating computations on sparse tensors and matrices using heterogeneous parallel architectures

- **Software** (**open-source**):
  - develop tensor computation libraries
  - contribute to improving the performance of tensor computations in machine learning libraries like PyTorch and TensorFlow

Efficient Large-Scale Sparse Multilinear Algebra Computations

- Parallel tensor contraction on accelerators (Long Term)
- Parallel conversion between sparse tensor layouts in memory (Short / Mid Term)
- Algorithms for hypergraphs in the language of multilinear algebra (Exploratory)



Parallel tensor contraction on accelerators (1/3)

- **Motivation**
  - Sequence of sparse tensor contractions arise in training and inference of neural networks
  - Tensor decomposition involves tensor contraction, where one of the operands is sparse
- This project will complement
  - the ongoing efforts at Topal for developing efficient methods for training and inference of deep neural networks
  - the planned extension of the **Chameleon** linear algebra library to support tensor computations in the context of the **PEPR NumPEx** project
- **Involved Topal members**: Olivier Beaumont, Lionel Eyraud-Dubois, Julia Gusak, Pierre Ramet, Mathieu Faverge, Pierre Estérie (NumPEx)

Parallel tensor contraction on accelerators (2/3)

- Contraction of two tensors: $\mathcal{A} \in \mathbb{R}^{I \times J \times P \times Q}$ and $\mathcal{B} \in \mathbb{R}^{P \times Q \times K \times L}$ with p,q as contraction indices

for *nonzero* $\mathcal{A}[i,j,p,q]$ ***in*** *subtensor* $\mathcal{A}[i,j,:,:]$ **do**
    for *nonzero* $\mathcal{B}[p,q,k,l]$ ***in*** *subtensor* $\mathcal{B}[p,q,:,:]$ **do**
        $\mathcal{C}[i,j,k,l] \mathrel{+}= \mathcal{A}[i,j,p,q] \cdot \mathcal{B}[p,q,k,l]$ `// requires lookup and insert on` $\mathcal{C}$

Challenges

- Contraction of sparse tensors involves high data movement and irregular algorithms
- Design space for finding the optimal configuration for minimizing data movement is huge!
  - sparsity pattern and number of nonzeros of input and output tensors: known at runtime
  - memory layout (data structure) of input and output tensors: several possibilities
- In sequence of tensor contractions, tensor contraction order is crucial!
  - affects memory footprint of intermediate results, and number of operations

Parallel tensor contraction on accelerators (3/3)

Approach

- Design architecture-aware methods for parallel tensor contraction for GPU
  - exploit GPU's massive multithreading and high memory bandwidth
- Minimize data movement through GPU's memory hierarchy
  - explore tiling of sparse tensors, and reordering and renumbering of sparse tensors
  - design architecture and input aware performance models to analyze data movement
- Design efficient parallel dynamic data structures (hash map, set, . . .) for GPUs
  - required for efficient sparse tensor computations, graph processing
  - parallel dynamic data structures for GPU have received limited attention
  - develop hashing-based methods for sparse tensor contraction on GPU

Parallel conversion between sparse tensor layouts

Motivation

- Several memory layouts (data structures) exist for sparse tensors
  - several tensor layouts (data structures) possible due to large number of dimensions
  - different layouts optimal for different operations and sparsity pattern
- Conversion of layouts lies on the critical path of computation for a sequence of operations

Approach

- Parallelize conversion of tensor layouts on accelerators, such as GPU
  - reduce cost of data movement, and exploit the high memory bandwidth of GPU
- Characterize tensor layouts, and develop models to analyze and predict the best layout
- **Involved Topal members**: Pierre Ramet, Abdou Guermouche and Mathieu Faverge

Further opportunities for contributions at Topal

- Inference on compressed sparse neural networks
  - involves performance-accuracy tradeoff
  - prior experience with inexact computation on graphs and with managing performance-accuracy tradeoff
- Contribute to enhancing the performance of **PaStiX** sparse linear solver
  - prior experience with high performance graph computation (graph $\Leftrightarrow$ sparse matrix)

Service

- Reviewer for journals [over **5**]: IEEE TPDS (2021), ACM TACO (2022), ...
- Reviewer for conferences and workshops: GrAPL 2023 (IPDPSW), ECOOP (2023, 2022)
- Artifact-evaluation committee [over **10**]: PPoPP (2021, 2018), PLDI (2024, 2022), ...

Selected Software

- Algorithms and Data Structures for Hyperedge Queries
  - ∼3K LOC, C++ (open-source)
  - **ACM Results Reproduced** badge
  - **State-of-the-art** for the hyperedge-existence query problem
- Optimize and Integrate Standalone Tracking Library (SixTrackLib)
  - ∼13K LOC, C/C++, OpenCL (open-source)
  - Contributed code largely **integrated into the SixTrackLib library maintained by and used at CERN**
- Efficient Parallel Sparse Tensor Contraction
  - ∼4.5K LOC, C++
  - To be released soon...

Backup Slides

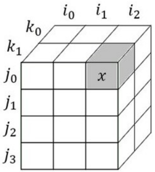
Input Tensors

Tensor	d	Dimensions	n
chicago_crime (T-1)	4	6,186 × 24 × 77 × 32	5,330,673
vast-2015-mc1-3d (T-2)	3	165,427 × 11,374 × 2	26,021,854
vast-2015-mc1-5d (T-3)	5	165,427 × 11,374 × 2 × 100 × 89	26,021,945
enron (T-4)	4	6,066 × 5,699 × 244,268 × 1,176	54,202,099
nell-2 (T-5)	3	12,092 × 9,184 × 28,818	76,879,419
flickr-3d (T-6)	3	319,686 × 28,153,045 × 1,607,191	112,890,310
flickr-4d (T-7)	4	319,686 × 28,153,045 × 1,607,191 × 731	112,890,310
delicious-3d (T-8)	3	532,924 × 17,262,471 × 2,480,308	140,126,181
delicious-4d (T-9)	4	532,924 × 17,262,471 × 2,480,308 × 1,443	140,126,181
nell-1 (T-10)	3	2,902,330 × 2,143,368 × 25,495,389	143,599,552

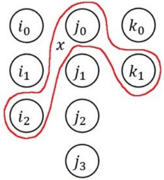
Input tensors from FROSTT dataset

Hyperedge Queries in Hypergraphs

- Tensors are multi-dimensional arrays
- A d-dimensional sparse tensor corresponds to a special class of hypergraphs



3D Tensor



Hypergraph

The problem

Given: A d-dimensional sparse tensor $\mathcal{T}$

Goal: Answer queries of the form: "Is $\mathcal{T}[i_1, \ldots, i_d]$ zero or nonzero?"

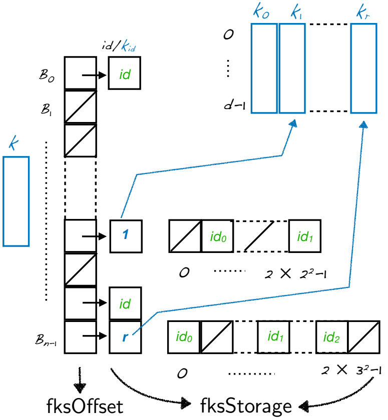
Motivation: *Tensor decomposition*[§]: used for finding patterns in massive data

---

§ T. G. Kolda and D. Hong, "Stochastic gradients for large-scale tensor decomposition," SIAM Journal on Mathematics of Data Science 2(2020)

Hyperedge Queries in Hypergraphs

Our proposal: Space-efficient hashing-based method with worst-case $O(1)$ lookups



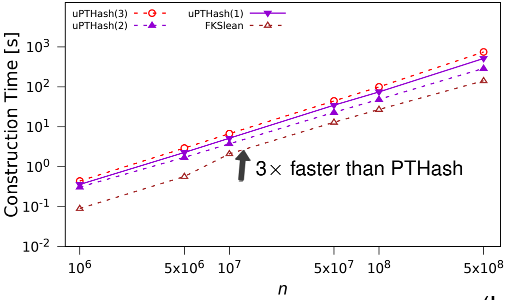
Hash data structure

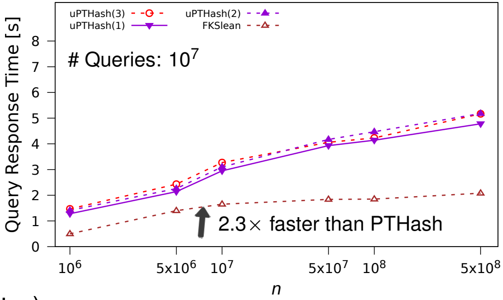
- All nonzeros are available at the start
- All nonzeros have unique coordinates
- A two-level hash data structure
- First level hash function: $h(\mathbf{k}, \mathbf{x}, p, n)$
  > Map nonzeros to buckets
- Second level hash function: $h(\mathbf{k}_i, \mathbf{x}, p, 2b_i^2)$
  > Determine unique positions for all nonzeros in each bucket

J. Bertrand, F. Dufossé, **S. Singh** and B. Uçar, "Algorithms and Data Structures for Hyperedge Queries", ACM Journal of Experimental Algorithms (JEA), vol. 27, no. 1, Article 1.13, 23 pages, 2022 [**ACM Results Reproduced Badge**]

Highlights and Results

- Guaranteed constant time lookup per query in the worst-case
- Construction time is linear in the number of nonzeros, in expectation
- Total storage space $O(n)$; less than $5n$ in practice
- **Fastest** among all the competitors, including the **state-of-the-art PTHash**





(Lower is better)

Past Work-I: GPU-specific approx. computing for parallel graph processing

Focus: Propagation-based graph algorithms following the vertex-centric model of parallelization

Method: Graph transformation techniques to make graphs amenable to parallel processing on GPU

- vertex renumbering and selective vertex replication
- controlled addition of edges

- Provide **tunable knobs** to control the performance-accuracy trade-off

1. Improve locality of graph data in memory
2. Exploit fast memory (e.g. cache) in the memory hierarchy
3. Mitigate load-imbalance among threads executing in parallel

S. Singh and R. Nasre, "Graffix: Efficient Graph Processing with a Tinge of GPU-Specific Approximations", 49th International Conference on Parallel Processing (ICPP) 2020, pp. 23:1–23:11