Scalable Tensor Algorithms for Modern Computing Systems

Suraj Kumar

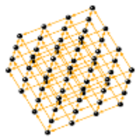
CNRS LIP/LaBRI Applicant

March 3, 2022

My Past Experience

Parallelization in Polyhedral Model (IISc, 2012)

- Linked-list operations
- Improved spatial locality
- Parallelization using OpenMP



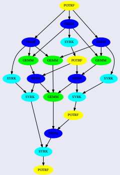
Seismic Imaging on GPU (IBM,2013)

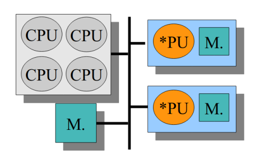
$$\begin{aligned}
H_1 &= \sin^2\theta\cos^2\phi\,\frac{\partial^2}{\partial x^2} + \sin^2\theta\sin^2\phi\,\frac{\partial^2}{\partial y^2} \\
&\quad + \cos^2\theta\,\frac{\partial^2}{\partial z^2} + \sin^2\theta\sin 2\phi\,\frac{\partial^2}{\partial x\partial y} \\
&\quad + \sin 2\theta\sin\phi\,\frac{\partial^2}{\partial y\partial z} + \sin 2\theta\cos\phi\,\frac{\partial^2}{\partial x\partial z} \\
H_2 &= \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} - H_1
\end{aligned}$$

Schedulers for Blue Gene Supercomputers (IBM, 2013)

- GASNET API
- Unbalanced Tree Search benchmark
- Comparison to Charm++

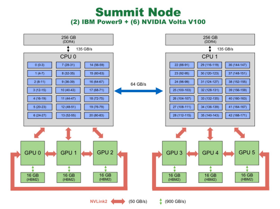
Scheduling on Heterogeneous Platforms (Inria, 2017)





Molecular Simulations on Supercomputers (PNNL, 2019)

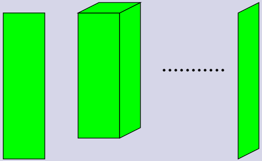
- NWChemEx Project
- TAMM library
- Hartree Fock and CCSD applications



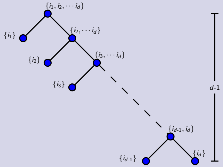
My Past Experience

Parallel Tensor Train Approximation (Inria, current)

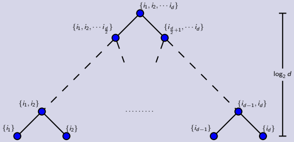
Small object representation



Sequential algorithm

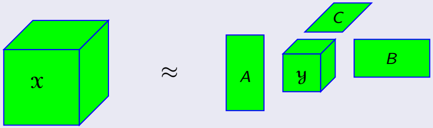


For better parallelization


$\{i_1, i_2, \cdots i_d\}$
$\{i_1, i_2, \cdots i_{\frac{d}{2}}\}$
$\{i_{\frac{d}{2}+1}, \cdots i_d\}$
$\{i_1, i_2\}$
$\{i_1\}$
$\{i_2\}$
....
$\{i_{d-1}, i_d\}$
$\{i_{d-1}\}$
$\{i_d\}$
$\log_2 d$

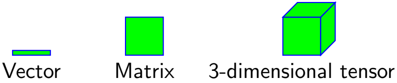
Communication Optimal Parallel Algorithms for Tensor Computations (Inria, current)

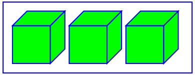
- Obtain $\mathcal{Y}$ from $\mathcal{X}, A, B, C$
- Obtain $\mathcal{X}$ from $\mathcal{Y}, A, B, C$



Tensors are used in Several Domains

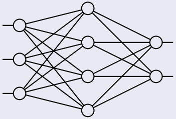
- **Neuroscience**: Neuron × Time × Trial
- **Transportation**: Pickup × Dropoff × Time
- **Media**: User x Movie x Time
- **Ecommerce**: User x Product x Time
- **Cyber-Traffic**: IP x IP x Port x Time
- **Social-Network**: Person x Person x Time x Interaction-Type




4-dimensional tensor

High Dimensional Tensors

- **Neural Network**:



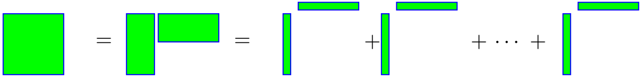
- **Molecular Simulation**: To represent wave functions
- **Quantum Computing**: Tensor network based models for computations

Tensor Computations

- Memory and computation requirements are exponential in the number of dimensions
  - A simulation involving just 100 spatial orbitals manipulates a huge tensor with 4^{100} elements

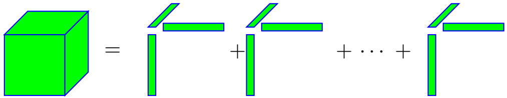
- People work with low dimensional structure (decomposition) of the tensors
  - A tensor is represented with smaller objects
  - Improves memory and computation requirements

- Most tensor decompositions rely on Singular Value Decomposition (SVD) of matrices
  - SVD represents a matrix as the sum of rank one matrices, $A = U\Sigma V^T = \sum_i \Sigma(i;i) U_i V_i^T$

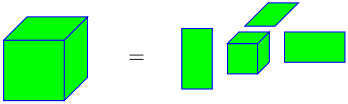


Popular Tensor Decompositions (Higher Order Generalization of SVD)

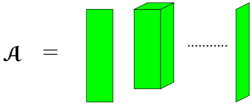
- Canonical decomposition (Also known as Canonical Polyadic or CANDECOMP/PARAFAC)



- Tucker decomposition

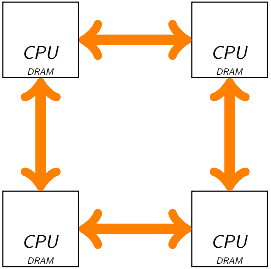


- Tensor Train decomposition (equivalently known as Matrix Product States)



Communication and its importance in HPC

- Running time of an algorithm depends on
  - Computations
    - Number of operations * time-per-operation
  - Data movement
    - Volume of communication / Network-bandwidth
    - Number of messages * Network-latency

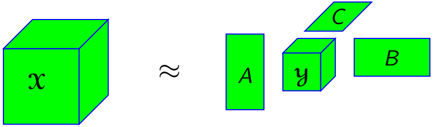


- Gaps growing exponentially with time (Source: Getting up to speed: The future of supercomputing)

	time-per-operation	Network-bandwidth	Network-latency
Annual improvements	59 %	26 %	15 %

- Avoid communication to save time (and energy)

Higher-order SVD (HOSVD) to compute Tucker decomposition



Algorithm 1 HOSVD Algorithm($\mathcal{X}$, R_1, R_2, R_3)

```
$A \leftarrow R_1$ left singular vectors of $\mathcal{X}_{(1)}$
$B \leftarrow R_2$ left singular vectors of $\mathcal{X}_{(2)}$
$C \leftarrow R_3$ left singular vectors of $\mathcal{X}_{(3)}$
$\mathcal{Y} = \mathcal{X} \times_1 A^{\mathsf{T}} \times_2 B^{\mathsf{T}} \times_3 C^{\mathsf{T}}$
Return $\mathcal{Y}$, $A$, $B$, $C$
```

- $\mathcal{X}$, $\mathcal{Y}$: 3-dimensional input and output tensors (or arrays) & A, B, C: matrices
- $\mathcal{X}_{(i)}$: matricization of $\mathcal{X}$ (ith dimension represents rows and remaining dimensions represent columns)
- Multiple Tensor-Times-Matrix (Multi-TTM) computation: $\mathcal{Y} = \mathcal{X} \times_1 A^{\mathsf{T}} \times_2 B^{\mathsf{T}} \times_3 C^{\mathsf{T}}$
  - To obtain full tensor, $\mathcal{X} = \mathcal{Y} \times_1 A \times_2 B \times_3 C$

Lower Bounds and Communication Optimal Algorithms

1. For Matrix Matrix Multiplications
2. For Multi-TTM Computation

Assumptions

- P number of processors
- Each processor performs (asymptotically) equal amount of operations
- No redundant operations
- One copy of data is in the system
  - $1/P$th amount of inputs (before the computation) and output (after the computation) on each processor
- Each processor has enough memory

This is joint work with Laura Grigori (Inria Paris, France), Grey Ballard (Wake Forest University, USA), Kathryn Rouse (Inmar Intelligence, USA), and Hussam Al Daas (Rutherford Appleton Laboratory, UK).

Existing Lower Bounds for Matrix Matrix Multiplications

- $C = AB$, where $A \in \mathbb{R}^{n_1 \times n_2}$, $B \in \mathbb{R}^{n_2 \times n_3}$, and $C \in \mathbb{R}^{n_1 \times n_3}$
- Let $d_1 = \min(n_1, n_2, n_3) \leq d_2 = median(n_1, n_2, n_3) \leq d_3 = \max(n_1, n_2, n_3)$

Existing Communication Lower Bounds (CARMA [IPDPS 2013])

P

1 — $2\frac{d_3}{d_2}$ — $2\frac{d_2 d_3}{d_1^2}$

LB=$\mathcal{O}(d_1 d_2)$

$\mathcal{O}((\frac{d_1^2 d_2 d_3}{P})^{1/2})$

$\mathcal{O}((\frac{d_1 d_2 d_3}{P})^{2/3})$

Our Communication Lower Bounds

P

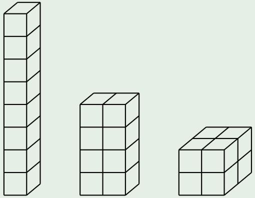
1 — $\frac{d_3}{d_2}$ — $\frac{d_2 d_3}{d_1^2}$

LB=$d_1 d_2 - \frac{d_1 d_2}{P}$

LB=$2(\frac{d_1^2 d_2 d_3}{P})^{1/2} - \frac{d_1 d_2 + d_1 d_3}{P}$

LB=$3(\frac{d_1 d_2 d_3}{P})^{2/3} - \frac{d_1 d_2 + d_1 d_3 + d_2 d_3}{P}$

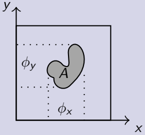
Arrangements of 8 processors



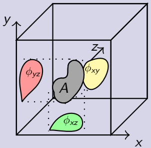
Loomis-Whitney & Hölder-Brascamp-Lieb inequalities

Size of $d-1$ dimensional projections (Loomis-Whitney inequalitiy)

- 2-dimensional object A and its 1-dimensional projections ϕ_x, ϕ_y
- $\phi_x \phi_y \geq Area(A)$



- 3-dimensional object A and its 2-dimensional projections: ϕ_{xy}, ϕ_{yz}, ϕ_{xz}
- $(\phi_{xy}\phi_{yz}\phi_{xz})^{\frac{1}{3-1}} \geq Volume(A)$



Hölder-Brascamp-Lieb (HBL) inequality – Generalization of Loomis-Whitney inequality

$$\mathbf{\Delta} = \begin{array}{c} \\ i \\ j \\ k \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

for $i = 1{:}n_1$, for $k = 1{:}n_2$, for $j = 1{:}n_3$

$C[i][j] += A[i][k] * B[k][j]$

- Find $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ such that $\mathbf{\Delta}.\mathbf{x} \geq \mathbf{1}$, $\mathbf{1}$ is vector of all ones
- ϕ_A, ϕ_B, ϕ_C: projections of computations on arrays A, B, C
- HBL inequality: $\phi_A^{x_1}\phi_B^{x_2}\phi_C^{x_3} \geq$ Amount of computations
- To make inequality tight select $\mathbf{x}$ such that $\mathbf{1}^T\mathbf{x}$ is minimum $=> x_1 = x_2 = x_3 = \frac{1}{2}$

Constraints for Matrix Multiplications

$$\text{for } i = 1{:}n_1, \text{ for } k = 1{:}n_2, \text{ for } j = 1{:}n_3$$
$$C[i][j] += A[i][k] * B[k][j]$$

- Total number of multiplications $= n_1 n_2 n_3$
- Consider a processor which performs $\frac{n_1 n_2 n_3}{P}$ amount of multiplications
- Optimization problem:

$$\textit{Minimize } \phi_A + \phi_B + \phi_C \text{ s.t.}$$
$$\phi_A^{\frac{1}{2}} \phi_B^{\frac{1}{2}} \phi_C^{\frac{1}{2}} \geq \frac{n_1 n_2 n_3}{P}$$

Extra constraints (our contributions)

- Each element of A (resp. B) is involved in n_3 (resp. n_1) multiplications
  - To perform at least $\frac{n_1 n_2 n_3}{P}$ multiplications: $\phi_A \geq \frac{n_1 n_2}{P}, \phi_B \geq \frac{n_2 n_3}{P}$
- Each element of C is the sum of n_2 multiplications, therefore $\phi_C \geq \frac{n_1 n_3}{P}$
- Projections can be at max the size of the arrays: $\phi_A \leq n_1 n_2, \phi_B \leq n_2 n_3, \phi_C \leq n_1 n_3$

Optimization Problem to Compute Communication Lower Bounds

- Projections (ϕ_A, ϕ_B, ϕ_C) indicate the amount of array access
- Communication lower bound $= \phi_A + \phi_B + \phi_C -$ data owned by the processor

$$\begin{aligned} &\textit{Minimize } \phi_A + \phi_B + \phi_C \ \text{ s.t.} \\ &\phi_A^{\frac{1}{2}} \phi_B^{\frac{1}{2}} \phi_C^{\frac{1}{2}} \geq \frac{n_1 n_2 n_3}{P} \\ &\frac{n_1 n_2}{P} \leq \phi_A \leq n_1 n_2 \\ &\frac{n_2 n_3}{P} \leq \phi_B \leq n_2 n_3 \\ &\frac{n_1 n_3}{P} \leq \phi_C \leq n_1 n_3 \end{aligned}$$

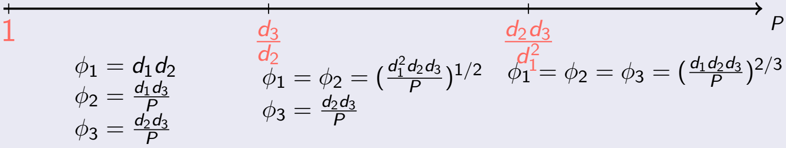
Generalized version (in terms of d_1, d_2, d_3)

$$\begin{aligned} &\textit{Minimize } \phi_1 + \phi_2 + \phi_3 \ \text{ s.t.} \\ &\phi_1^{\frac{1}{2}} \phi_2^{\frac{1}{2}} \phi_3^{\frac{1}{2}} \geq \frac{d_1 d_2 d_3}{P} \\ &\frac{d_1 d_2}{P} \leq \phi_1 \leq d_1 d_2 \\ &\frac{d_1 d_3}{P} \leq \phi_2 \leq d_1 d_3 \\ &\frac{d_2 d_3}{P} \leq \phi_3 \leq d_2 d_3 \\ &d_1 \leq d_2 \leq d_3 \end{aligned}$$

Amount of Accesses and Communication Lower bounds

- Estimate the solution based on Lagrange multipliers
- Prove optimality using all KarushKuhnTucker (KKT) conditions are satisfied

Amount of accesses $=\phi_1 + \phi_2 + \phi_3$

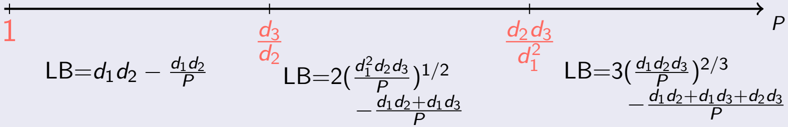


$\phi_1 = d_1 d_2$
$\phi_2 = \frac{d_1 d_3}{P}$
$\phi_3 = \frac{d_2 d_3}{P}$

$\phi_1 = \phi_2 = (\frac{d_1^2 d_2 d_3}{P})^{1/2}$
$\phi_3 = \frac{d_2 d_3}{P}$

$\phi_1 = \phi_2 = \phi_3 = (\frac{d_1 d_2 d_3}{P})^{2/3}$

Communication Lower Bounds (Amount of Data Transfers)



$\text{LB}=d_1 d_2 - \frac{d_1 d_2}{P}$

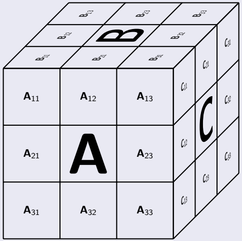
$\text{LB}=2(\frac{d_1^2 d_2 d_3}{P})^{1/2} - \frac{d_1 d_2 + d_1 d_3}{P}$

$\text{LB}=3(\frac{d_1 d_2 d_3}{P})^{2/3} - \frac{d_1 d_2 + d_1 d_3 + d_2 d_3}{P}$

Design of Communication Optimal Algorithms

Data Distribution (P is organized into a $p_1 \times p_2 \times p_3$ grid)

- Select p_1, p_2, and p_3 based on the lower bounds
- Each processor has $\frac{1}{P}$th amount of A, B and C
- $A_{11} = A(1:\frac{n_1}{p_1}, 1:\frac{n_2}{p_2})$ is evenly distributed among $(1, 1, *)$ processors
- Similar data distributions for B and C



Algorithm 2 $C = AB$ Matrix Multiplication Algorithm

```
$(p_1', p_2', p_3')$ is my processor id
//All-gather input matrices $A$ and $B$
$A_{p_1' p_2'}$ = All-Gather($A$, $(p_1', p_2', *)$)
$B_{p_2' p_3'}$ = All-Gather($B$, $(*, p_2', p_3')$)
$T$ = Local-Matrix-Multiplication($A_{p_1' p_2'}$, $B_{p_2' p_3'}$) // Local matrix multiplication in a temporary
Reduce-Scatter($C_{p_1' p_3'}$, T, $(p_1', *, p_3')$) // Reduce-scatter the output
```

3-dimensional Multi-TTM ($\mathcal{Y} = \mathcal{X} \times_1 \mathbf{A}^{(1)^\mathsf{T}} \times_2 \mathbf{A}^{(2)^\mathsf{T}} \times_3 \mathbf{A}^{(3)^\mathsf{T}}$)

- TTM-in-Sequence approach (used in Tucker-MPI)
    - For 2-dimensional computation, $\mathbf{Y} = \mathbf{A}^{(1)\mathsf{T}}\mathbf{X}\mathbf{A}^{(2)}$

All-at-Once approach (our contribution)

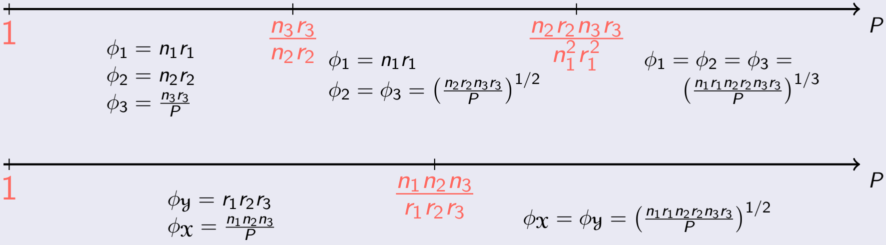
```
for $n_1' = 1{:}n_1$, for $n_2' = 1{:}n_2$, for $n_3' = 1{:}n_3$
  for $r_1' = 1{:}r_1$, for $r_2' = 1{:}r_2$, for $r_3' = 1{:}r_3$
    $\mathcal{Y}(r_1', r_2', r_3') = \mathcal{Y}(r_1', r_2', r_3')$
        $+ \left(\mathcal{X}(n_1', n_2', n_3') * \mathbf{A}^{(1)}(n_1', r_1') * \mathbf{A}^{(2)}(n_2', r_2') * \mathbf{A}^{(3)}(n_3', r_3')\right)$
```

$$\mathbf{\Delta} = \begin{bmatrix} \mathbf{I}_{3\times 3} & \mathbf{1}_3 & \mathbf{0}_3 \\ \mathbf{I}_{3\times 3} & \mathbf{0}_3 & \mathbf{1}_3 \end{bmatrix}$$

- Total number of inner $(4 - \mathit{array})$ operations $= n_1 r_1 n_2 r_2 n_3 r_3$
- $\mathbf{\Delta}$ is not full rank: allows us to get multiple constraints related to computations
- Possible to solve matrix and tensor optimization problems separately

Amount of Accesses and Lower bounds

Amount of accesses $= \phi_x + \phi_y + \phi_1 + \phi_2 + \phi_3$



1 $\qquad \frac{n_3 r_3}{n_2 r_2}$ $\qquad \frac{n_2 r_2 n_3 r_3}{n_1^2 r_1^2}$ $\qquad P$

$$\phi_1 = n_1 r_1 \quad \phi_2 = n_2 r_2 \quad \phi_3 = \frac{n_3 r_3}{P}$$

$$\phi_1 = n_1 r_1 \quad \phi_2 = \phi_3 = \left(\frac{n_2 r_2 n_3 r_3}{P}\right)^{1/2}$$

$$\phi_1 = \phi_2 = \phi_3 = \left(\frac{n_1 r_1 n_2 r_2 n_3 r_3}{P}\right)^{1/3}$$

1 $\qquad \frac{n_1 n_2 n_3}{r_1 r_2 r_3}$ $\qquad P$

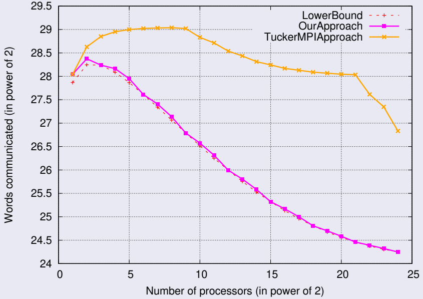
$$\phi_y = r_1 r_2 r_3 \quad \phi_x = \frac{n_1 n_2 n_3}{P}$$

$$\phi_x = \phi_y = \left(\frac{n_1 r_1 n_2 r_2 n_3 r_3}{P}\right)^{1/2}$$

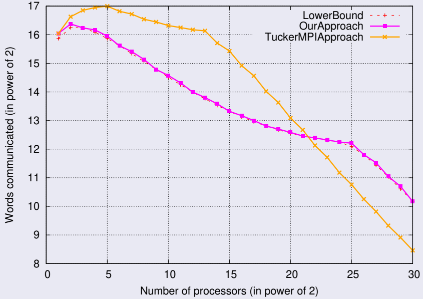
- We assume $n_1 r_1 \leq n_2 r_2 \leq n_3 r_3$ and $r_1 r_2 r_3 \leq n_1 n_2 n_3$
- Communication lower bound $= \phi_x + \phi_y + \phi_1 + \phi_2 + \phi_3 - \frac{n_1 n_2 n_3 + r_1 r_2 r_3 + n_1 r_1 + n_2 r_2 + n_3 r_3}{P}$
- Can design similar algorithm (though 6-dimensional)for this
- Selection of optimal processor grid dimensions based on the lower bound requires some adaption

Simulated Performance Comparison of Our Algorithm

$n_1 = n_2 = n_3 = 2^{20}, r_1 = r_2 = r_3 = 2^8$



$n_1 = n_2 = n_3 = 2^{12}, r_1 = r_2 = r_3 = 2^4$



- Typical scenarios in data compression problems
- Lower Bound is only valid for our approach
- For $P << \frac{n_1 n_2 n_3}{r_1 r_2 r_3}$, our approach communicates much less than the state-of-the-art approach (TuckerMPI)

Project: Scalable Tensor Algorithms for Modern Computing Systems

1. Design of Scalable Communication Optimal Algorithms for Tensors (Main Focus)
2. Extension of Existing Approaches/Algorithms (Short/Mid Term Research Plans)
3. Exploratory Topics (Mid/Long Term Research Plans)

Scalable communication optimal algorithms for tensors

- Analyze existing algorithms
- Determine communication lower bounds
- Propose communication optimal algorithms
- Implement the proposed algorithms

Main focus

Extension of existing approaches

- Strassen's concepts to tensors
- Concepts of hierarchical matrices to tensors
- Separation order of dimensions in tensor train

Short/Mid term plans

Exploratory topics

- New tensor representations
- Architecture aware algorithms
- Randomization in tensors
- Factorizations of tensors

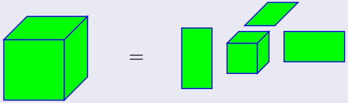
Mid/Long term plans

Scalabale algorithms for popular tensor operations

- Determine the communication lower bounds for tensor decompositions
- Analyse the popular decomposition algorithms and communications performed by them
- Propose new scalable communication optimal algorithms
  - If possible design tiles/tasks based algorithms
- Implement the proposed algorithms
  - Handle performance issues for homogeneous systems
    - Load balancing
    - Memory aware approaches
    - scheduling strategies
- Same for manipulation operations of popular tensor representations
- Extend implementation for heterogeneous systems (start with Nvidia GPUs based heterogeneous systems)
- Create a tensor library

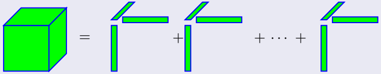
Popular tensor decompositions

Tucker decomposition



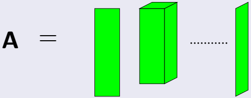
- Determine communication lower bounds for this operation
- Analyse communications performed by state of the art algorithms
- Propose and implement new scalable communication algorithms

Canonical decomposition



- No deterministic algorithm to find the decomposition
- Analyse one iteration of the popular existing algorithms
- Propose and implement scalable algorithms for one iteration

Tensor Train decomposition



- Determine communication lower bounds for this operation
- Analyse communication performed by popular algorithm
- Propose and implement new scalable communication algorithms

Proving communication lower bounds for parallel computations

How people did it for linear algebra operations?

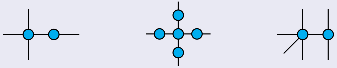
- People obtain results for matrix multiplication operations
- Same lower bounds apply to almost all direct linear linear algebra operations using reduction [Ballard et. al., 09] , for instance, bound for LU factorization

$$\begin{pmatrix} I & & -B \\ A & I & \\ & & I \end{pmatrix} = \begin{pmatrix} I & & \\ A & I & \\ & & I \end{pmatrix} \begin{pmatrix} I & & -B \\ & I & AB \\ & & I \end{pmatrix}$$

Approach to compute lower bounds for tensor computations

Notation: Tensors are denoted by solid shapes and number of lines denote the dimensions of the tensors. Connecting two lines implies summation (or contraction) over the connected dimensions.

- Obtain bounds for basic tensor operations: Tensor times matrix (TTM), Multiple tensor times matrix (Multi-TTM), Tensor contraction



- Express decompositions and manipulations in terms of these basis operations

Research Project

1. Design of Scalable Communication Optimal Algorithms for Tensors (Main Focus)
2. Extension of Existing Approaches/Algorithms (Short/Mid Term Research Plans)
3. Exploratory Topics (Mid/Long Term Research Plans)

Strassen's concepts to tensors

Matrix multiplication of $n \times n$ square matrices

- Complexity of traditional matrix multiplication is $\mathcal{O}(n^3)$
- Strassen's matrix multiplication
  - Expressed matrix multiplication as a tensor computation
  - Canonical rank of the tensor determines the complexity of the computation
  - Complexity is $\mathcal{O}(n^{2.81})$
- Plan to extend Strassen's concepts to tensor contractions

Contraction of a 3-dimensional tensor with a matrix

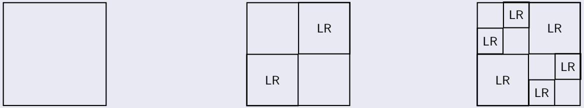
```
for $i_1 = 1 : n$ do
    for $i_2 = 1 : n$ do
        for $i_3 = 1 : n$ do
            for $j_2 = 1 : n$ do
                $\mathcal{G}(i_1, i_2, j_2) = \mathcal{G}(i_1, i_2, j_2) + \mathcal{A}(i_1, i_2, i_3) * B(i_3, j_2)$
            end for
        end for
    end for
end for
```

- Total $\mathcal{O}(n^4)$ operations
- Apply Strassen's algorithm for each i_1, total $\mathcal{O}(n^{3.81})$ operations
- Expressing as a canonical decomposition of $8 \times 8 \times 4$ tensor can further reduces the number of operations

Hierarchical Matrix concepts to Tensors

Hierarchical Matrices

- Data sparse approximation of non-sparse matrices

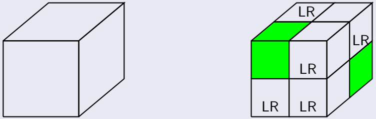


Original Matrix Step 1 Step 2

LR: low rank block

Tensors

- $f(i,j,k) = \frac{1}{|i-j|+|j-k|+|k-i|}$
- Value is small if difference of any pair is large
- Formalize and evaluate this approach for tensors



Original Tensor Step 1

Research Project

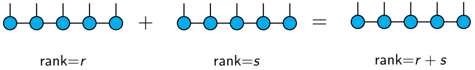
1. Design of Scalable Communication Optimal Algorithms for Tensors (Main Focus)

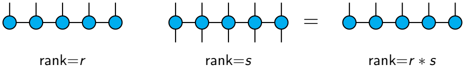
2. Extension of Existing Approaches/Algorithms (Short/Mid Term Research Plans)

3. Exploratory Topics (Mid/Long Term Research Plans)

Tensor Representations for High Dimensional Tensors

- Tensor Train is a popular representation to work with high dimensional tensors
- Adding tensors and aplying an operator in this representation

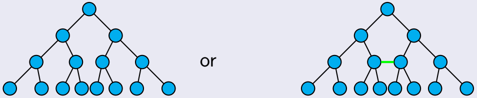




- Requires a truncation process which iterates over cores one by one
- This representation is not much suited to work in parallel

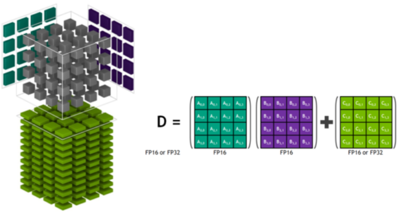
New Tensor Representations

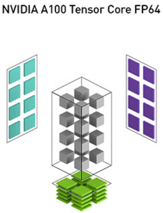
- Look at new representations in tree format – suitable for parallelization



  - Data will be stored at the leaf nodes
  - Interal nodes will help to manipulate tensors in parallel

Architecture Aware Algorithms




NVIDIA A100 Tensor Core FP64

- Recent Nvidia GPUs have tensor cores to accelerate AI computations
- Most linear algebra computations do not take advantages of these units
- Design algorithms which take architecture details into account

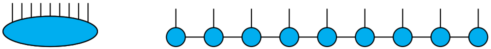
Fig source: `www.nvidia.com`

Randomization in Tensor Computations

- Randomized SVD and UTV factorization are now well established

- Apply randomization to tensors

- Perform factorizations of tensors
  - For example: QR like factorization of a tensor



Integration in the LIP/LaBRI laboratory

Communication optimal tensor algorithms

- Analyze existing algorithms
- Determine communication lower bounds
- Propose communication optimal algorithms
- Implement the proposed algorithms

Main focus

Extension of existing approaches

- Strassen's concepts to tensors
- Concepts of hierarchical matrices to tensors
- Separation order of dimensions in tensor train

Short/Mid term plans

Exploratory topics

- New tensor representations
- Architecture aware algorithms
- Randomization in tensors
- Factorizations of tensors

Mid/Long term plans

ROMA team (LIP laboratory)

- *Bora Ucar*: design of tensor compression and manipulation algorithms
- *Gregoire Pichon*: low-rank based algorithms
- *Anne Benoit, Loris Marchal, Yves Robert and Frederic Vivien*: scalability and scheduling aspects in the long term

SATANAS team (LaBRI laboratory)

- *Olivier Beaumont* and *Lionel Eyraud-Dubois*: tensor train based neural network models
- *Mathieu Faverge*: low-rank based methods
- *Abdou Guermouche, Samuel Thibault*: exploitation of maximum potential of HPC systems in the long term

Bringing additional skills in the team

- High dimensional dense tensor computations, use of tensors in molecular simulations
- Communication lower bounds for linear algebra computations
- Scalable approaches for large HPC systems