Scalable Tensor Algorithms for Modern Computing Systems

Suraj Kumar

Inria ROMA Applicant

June 7, 2021

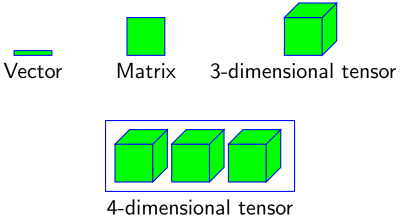
Resume

Interests: Tensors, Scalable Algorithms, Scheduling, Runtime Systems, Performance Optimizations

- Jul 2012 — **IBM Research, New Delhi, India**
  Software Engineer
  - Optimizations on GPUs
  - Schedulers for BG/P machine
- Dec 2013 — **Inria Bordeaux, France**
  PhD Student
  - Scheduling on heterogeneous resources
- Aug 2017 — **Ericsson, Bangalore, India**
  Senior Engineer
  - Use of remote GPUs in cloud
- May 2018 — **Pacific Northwest National Laboratory, USA**
  Postdoc
  - Runtime systems for tensor operations
- Nov 2019 — **Alpines team, Inria Paris, France**
  Postdoc
  - Parallel algorithms for tensors

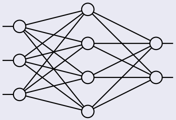
Tensors are used in Several Domains

- **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



High Dimensional Tensors

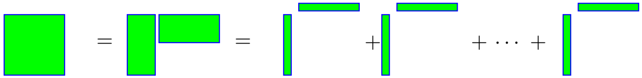
- **Neural Network**:



- **Molecular Simulation**: To represent wave functions
- **Quantum Computing**: To represent qubit states

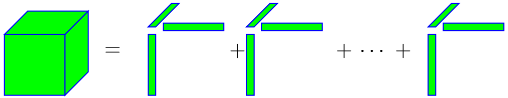
Tensor Computations

- 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, $A = U\Sigma V^T = \sum_i \Sigma(i;i) U_i V_i^T$

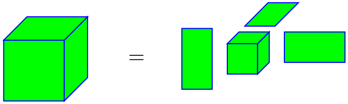


Popular Tensor Decompositions (Higher Order Generalization of SVD)

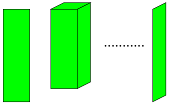
- Canonical decomposition (Also known as Canonical Polyadic or CANDECOMP/PARAFAC)



- Tucker decomposition



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

$\mathbf{A} =$ 

Tensor notation: bold letters denote tensors, i.e., $\mathbf{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is a d-dimensional tensor.

Previous Activities

1. Parallel Tensor Train Algorithms
2. Scheduling on Heterogeneous Systems
3. Minimize Impact of Data Transfers on Large Scale Systems

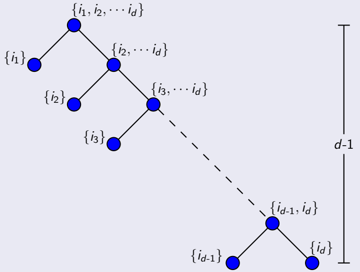
Other Significant Activities:

- Performance optimizations on GPUs
- Injecting static rules in dynamic schedulers
- Large scale runtime systems
- Communication lower bounds for computations

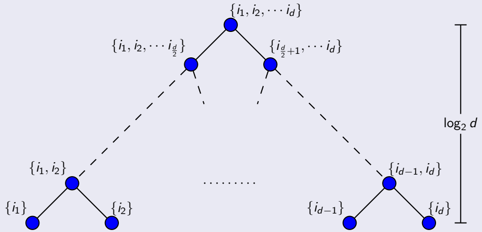
Tensor Train algorithms and Separation of dimensions

- A sequential algorithm to compute Tensor Train decomposition exists [Oseledets, 2011]

Sequential algorithm



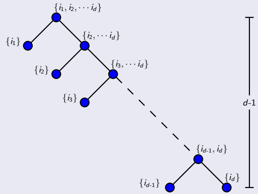
For better parallelization



- Can obtain better parallelism by expressing the operation in a balanced binary tree shape
    - Proposed a parallel algorithm based on this idea (joint work with L. Grigori, Inria Paris)

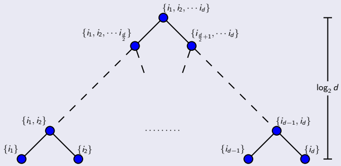
Tensor Train approximation algorithms

Sequential algorithm [Oseledets, 2011]



- Unfolding matrix: matricized representation of the tensor
- Perform truncated SVD of unfolding matrix A, $A = U\Sigma V^T + E_A$
- Work with ΣV^T on the right subtree

Our parallel algorithm



- Perform truncated SVD of unfolding matrix A, $A = U\Sigma V^T + E_A$
- Find diagonal matrices X, Y, and S, such that $\Sigma = XSY$
- Call left (resp. right) subtree with UX (resp. YV^T)

Approach 1: $X = I$, $Y = \Sigma$, $S = I$
Approach 2: $X = Y = \Sigma^{\frac{1}{2}}$, $S = I$
Approach 3: $X = Y = \Sigma$, $S = \Sigma^{-1}$

Comparison of our approaches

- A 12-dimensional tensor with 4^{12} elements (generated with a popular low rank function)
- prescribed accuracy $= 10^{-6}$
- Compr: compression ratio, NE: number of elements, AA: approximation accuracy

Metric	Sequential Algo	Parallel Algo		
		Approach 1	Approach 2	Approach 3
Compr	99.993	99.817	99.799	99.993
NE	1212	30632	33772	1212
AA	2.271e-07	3.629e-08	2.820e-08	2.265e-07

SVD is expensive

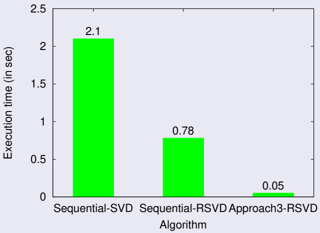
- Good alternatives to SVD: QR factorization with column pivoting (QRCP), randomized SVD (RSVD)

Approach	Rank	Compr	NE	Sequential-AA	Approach3-AA
SVD	5	99.994	992	6.079e-06	6.079e-06
QRCP+SVD				1.016e-05	1.384e-05
RSVD				6.079e-06	6.079e-06
SVD	6	99.992	1376	1.323e-07	1.340e-07
QRCP+SVD				3.555e-07	5.737e-07
RSVD				1.322e-07	1.322e-07

Performance Comparison

Single core performance

- Number of computations for both RSVD algorithms $= \mathcal{O}(n^d)$
- Approach3-SVD is very slow
- Approach3-RSVD is much faster

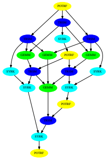


Parallel performance counts along the critical path on P processors

Algorithm	# Computations	Communications	# Messages
Sequential-RSVD	$\mathcal{O}(\frac{n^d}{P})$	$\mathcal{O}(\frac{n^{d-1}}{P})$	$\mathcal{O}(d \log P)$
Approach3-RSVD	$\mathcal{O}(\frac{n^d}{P})$	$\mathcal{O}(\frac{n^{\frac{d}{2}}}{\sqrt{P}} \log P)$	$\mathcal{O}(\log d \log P)$

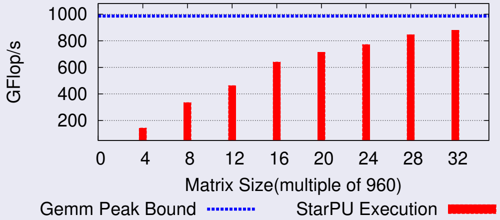
Scheduling on Heterogeneous Systems

- Heterogeneous systems are common in High Performance Computing (HPC) (147 out of 500 in TOP500 list)
- Task based runtimes are a popular approach to exploit these systems



- Task based runtimes: StarPU, OmpSS, Legion, PaRSEC
- Application is represented as a graph of tasks (computations)
- E.g., Cholesky graph for 4×4 tile matrix

StarPU scheduler performance



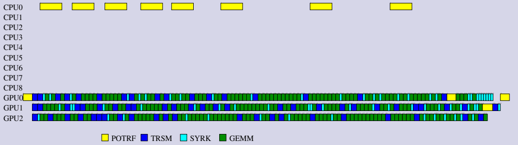
- A platform with 9 CPUs and 3 GPUs
- Scheduler is based on popular heft strategy

- Goal: Enhance performance bounds and propose better scheduling strategies

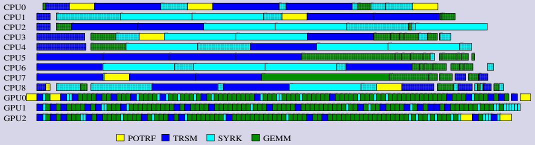
Joint work with E. Agullo, O. Beaumont, L. Eyraud-Dubois, and S. Thibault during my PhD at Inria Bordeaux

Our strategy and Performance comparison

Trace for 12 X 12 tile matrix of Cholesky factorization



StarPU scheduling strategy, performance = 686 GFlop/s

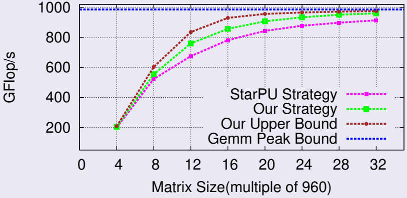


- Resource selects the best suited task
- Fast resource restarts the blocking task

Our strategy, performance = 760 GFlop/s

Theoretical guarantees of our strategy and performance comparison

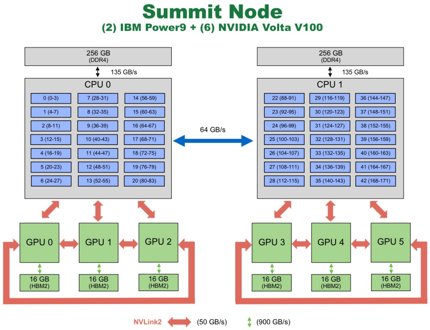
(#CPUs, #GPUs)	For a set of independent tasks: Approximation ratio	For a set of independent tasks: Worst case ex.
(1,1)	$\frac{1+\sqrt{5}}{2}$	$\frac{1+\sqrt{5}}{2}$
(m,1)	$\frac{3+\sqrt{5}}{2}$	$\frac{3+\sqrt{5}}{2}$
(m,n)	$2+\sqrt{2} \approx 3.41$	$2+\frac{2}{\sqrt{3}} \approx 3.15$

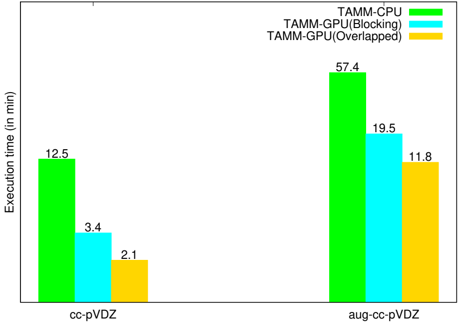


- Our upper bound is obtained by a linear program

Minimizing impact of communications on Summit supercomputer

- Maximizing the overlap of communications and computations
- Implemented proposed approaches in Tensor Algebra for Manybody Methods (TAMM) library
- Molecular chemistry application (CCSD), Ubiqtin molecule, cc-pVDZ (737 basis functions, 220 nodes), aug-cc-pVDZ (1243 basis functions, 256 nodes)





Joint work with S. Krishnamoorthy and M. Zalewski during my postdoc at PNNL, USA

Figure source: `https://www.olcf.ornl.gov`

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

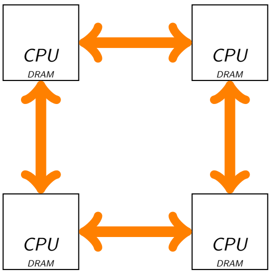
Exploratory topics

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

Mid/Long term plans

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 %

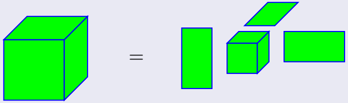
- Avoid communication to save time (and energy)

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

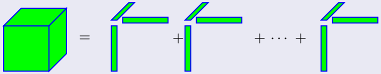
Popular tensor decompositions

Tucker decomposition



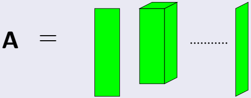
- Determine communication lower bounds for this operation
- Analyse communications performed by state of the art algorithms
- Propose and implement new scalable communication algorithms

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?

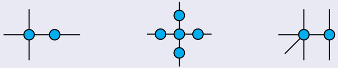
- People obtain results for matrix multiplication operations
- Same lower bounds apply to almost all direct linear linear algebra operations using reduction [Ballard et. al., 09] , for instance, bound for LU factorization

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

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

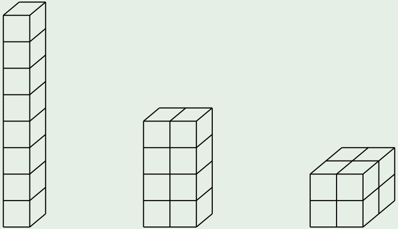


- Express decompositions and manipulations in terms of these basis operations

Communication lower bounds on P processors

- Recently started to work with L. Grigori (Inria Paris, France), H. Daas (Rutherford Appleton Laboratory, UK) and G. Ballard (Wake Forest University, USA)
- Revisited communication lower bounds for matrix multiplications
  - Expressed existing approaches in suitable forms for tensors
  - Lower bounds also instruct arrangement of processors in optimal algorithms
  - Improved the constants in the existing ranges of P (Demmel et.al [IPDPS 2013])

Arrangements of 8 processors



- Plan to continue this collaboration to compute lower bounds for tensor computations

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

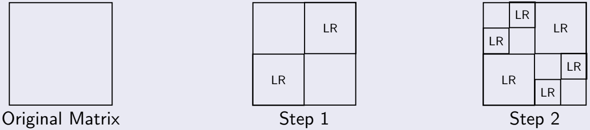
for $i_1 = 1 : n$ **do**
    for $i_2 = 1 : n$ **do**
        for $i_3 = 1 : n$ **do**
            for $j_2 = 1 : n$ **do**
                $\mathbf{G}(i_1, i_2, j_2) = \mathbf{G}(i_1, i_2, j_2) + \mathbf{A}(i_1, i_2, i_3) * B(i_3, j_2)$
            end for
        end for
    end for
end for

- 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

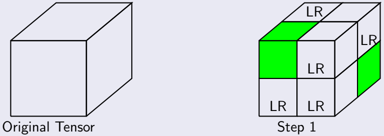
- Data sparse approximation of non-sparse matrices



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

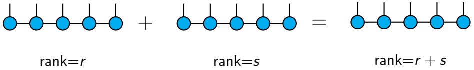


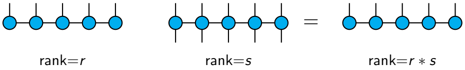
Research Project

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

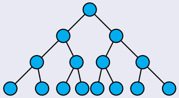




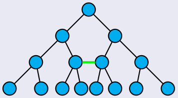
- Requires a truncation process which iterates over cores one by one
- This representation is not much suited to work in parallel

New Tensor Representations

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



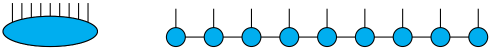
or



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

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 ROMA team

Anne Benoit, Loris Marchal, Gregoire Pichon, Yves Robert, Bora Ucar, and Frederic Vivien

Team

- Parallel algorithms
- Scheduling strategies
- Memory aware algorithms
- Sparse tensor computations
- Low rank compression algorithms

Expertise of the team

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

Bringing additional skills in the team

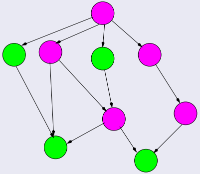
- High dimensional dense tensor computations
- Communication lower bounds for linear algebra computations
- Scalable approaches for large HPC systems
- Use of tensors in quantum and molecular simulations

Thank You!

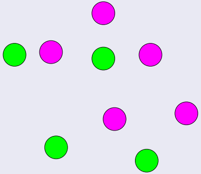
Backup Slides

Our Performance Bound

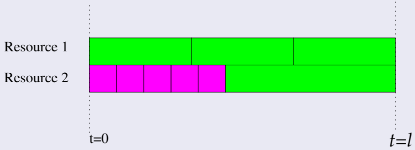
Task Graph



No Dependencies

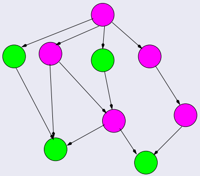


Minimum execution time (minimize l)

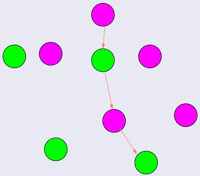


Our Performance Bound

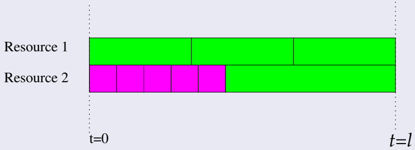
Task Graph



Some Dependencies



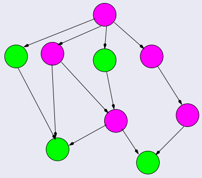
Minimum execution time (minimize l)



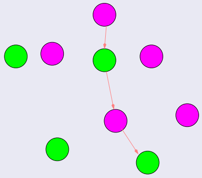
- ⋆ If any path in the graph is larger than l

Our Performance Bound

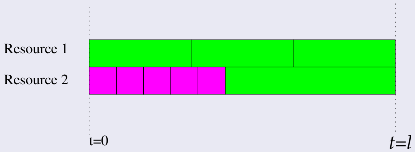
Task Graph



Some Dependencies



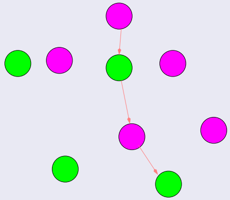
Minimum execution time (minimize l)



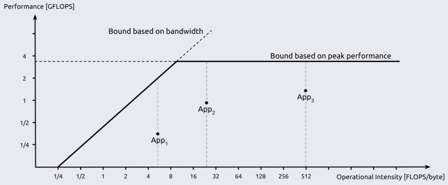
- ⋆ If any path in the graph is larger than l
  - add this path as a constraint and repeat the procedure

Roofline model with Dependencies

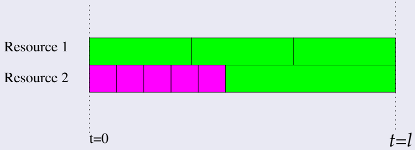
Our bound



Roofline Model



Minimum execution time (minimize l)

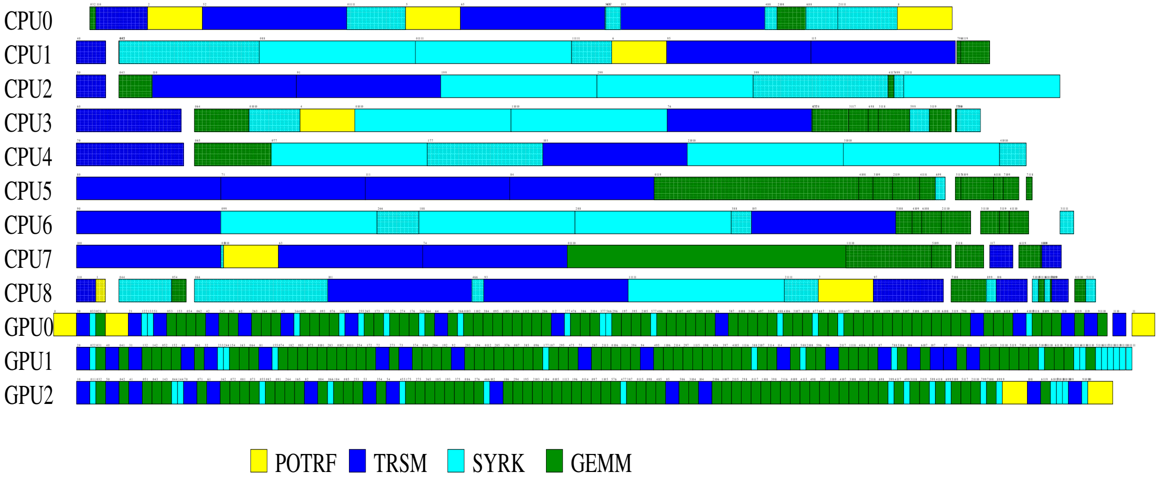


- ⋆ If any path in DAG is larger than l
  - add this path with data transfer cost as a constraint and repeat the procedure

- Take minimum of both values as the lower bound of the application

Our scheduling strategy

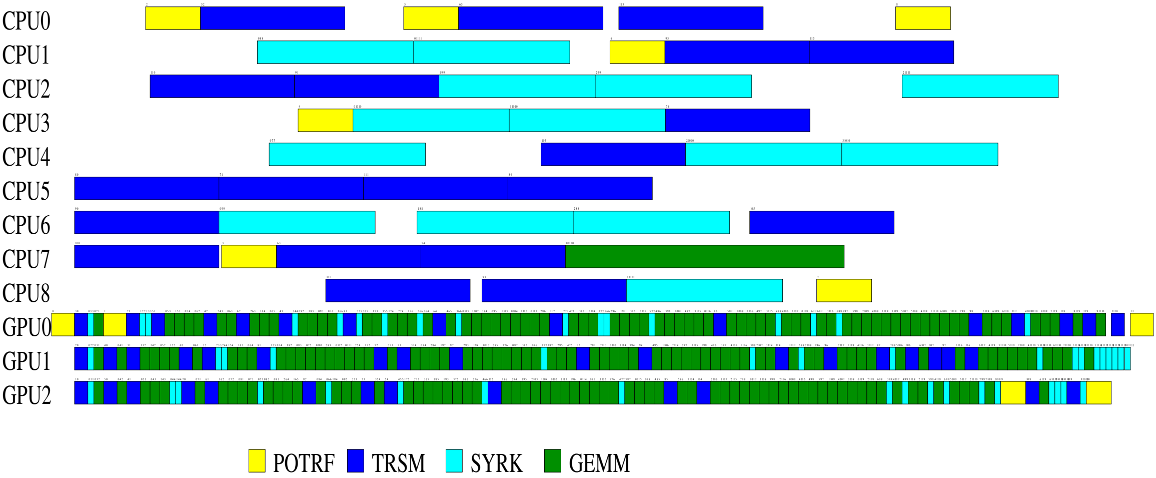
Trace for 12 X 12 tile matrix of Cholesky Factorization



Achieved performance = 760 GFlop/s, StarPU performance = 686 GFlop/s

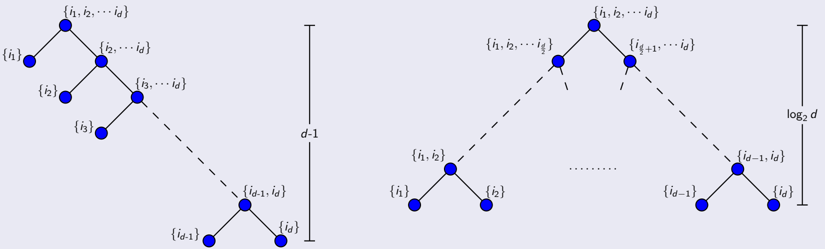
Our scheduling strategy

Trace for 12 X 12 tile matrix of Cholesky Factorization



Achieved performance = 760 GFlop/s, StarPU performance = 686 GFlop/s

Unfolding matrices of a tensor



k-th unfolding matrix of tensor $\mathbf{A}$ is defined as, $A_k = [A_k(i_1, i_2, \cdots, i_k; i_{k+1}, \cdots, i_d)]$

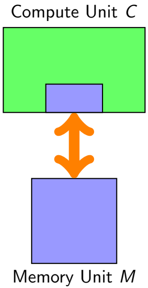
- Size of A_k is $(\prod_{l=1}^{k} n_l) \times (\prod_{l=k+1}^{d} n_l)$
- $r_k = \text{rank}(A_k)$
- Each node works with an unfolding matrix

Theorem: Our parallel algorithm produces a Tensor Train representation with ranks not higher than r_k.

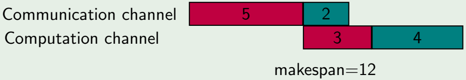
Communication Computation Overlap

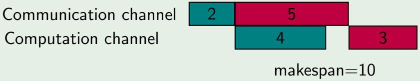
Task	Data Transfer Time	Computation Time
A	5	3
B	2	4

Problem: Given a set of tasks in what order we transfer them from M to C such that the makespan is minimal?



Possible schedules





Order of Data Transfers

- Compute intensive task: Computation time $\geq$ Data transfer time
- Communication intensive task: Computation time $<$ Data transfer time

Optimal Algorithm: When memory capacity of the compute unit is not a concern

- First sort compute intensive tasks in increasing order of their data transfer time
- Then sort the communication intensive tasks in decreasing order of their computation time

Memory capacity is limited

- Proved that the problem is NP-Complete
- Proposed static and dynamic based approaches and evaluated them on traces of molecular simulations
  - Static approaches: order is computed in advance
  - Dynamic approaches: next task is chosen based on the heuristic
  - Combination of both: start with static order and switch to dynamic based on available memory

Performance evaluation on Summit supercomputer

- Implemented static approaches in Tensor Algebra for Manybody Methods (TAMM) library
- CCSD application, Ubiqtin molecule, cc-pVDZ (737 basis functions, 220 nodes), aug-cc-pVDZ (1243 basis functions, 256 nodes)

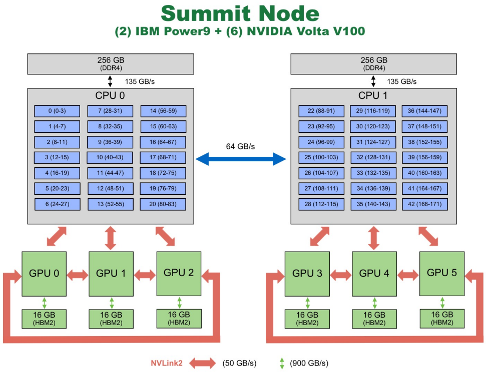
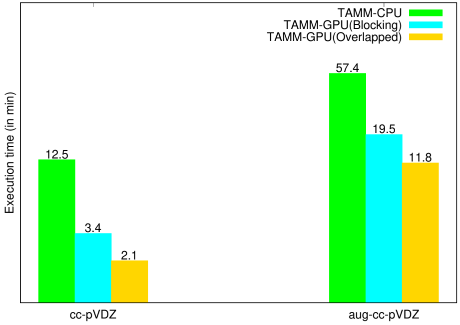
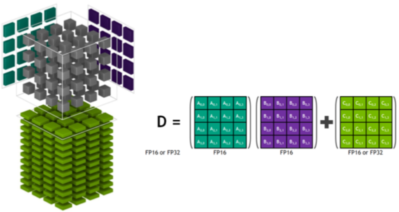
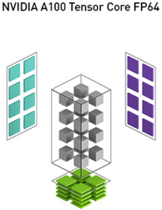


Figure source: `https://www.olcf.ornl.gov`



Architecture Aware Algorithms





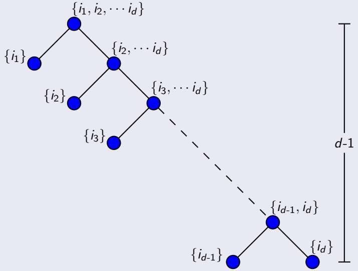
- Recent Nvidia GPUs have tensor cores to accelerate AI computations
- Most linear algebra computations do not take advantages of these units
- Design algorithms which take architecture details into account

Figure source: `www.nvidia.com`

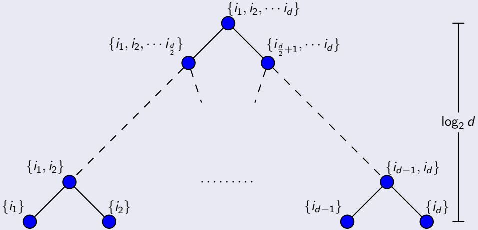
Order of separation of dimensions in tensor train algorithms

- Determine the order of separation of dimensions

Sequential algorithm



Our parallel algorithm



Communication lower bounds for Matrix Multiplications on P Processors

- Each processor asymptotically performs the same amount of computation
- One copy of data is distributed among processor

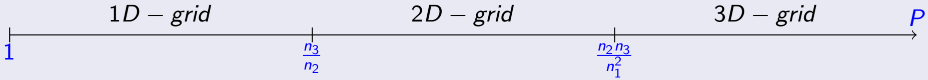
Square matrix multiplication

- Communication optimal algorithms consider 3-dimensional processor arrangement
- Rediscovered many times in literature

Rectangular matrix multiplication $C = AB$

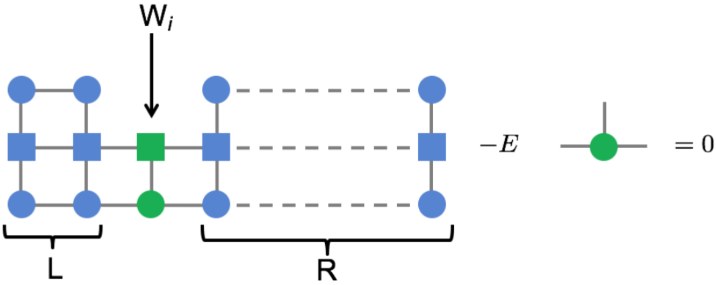
Assume $n_1 \leq n_2 \leq n_3$ and dimensions of A, B, and C are $n_1 \times n_2$, $n_2 \times n_3$ and $n_1 \times n_3$.

- Lower bounds depend on the dimensions of the matrices
- Requires to solve a linear program
- Lower bounds also instruct the arrangement of processors in optimal algorithms



- Improved the constants in the existing ranges of P (Demmel et.al [IPDPS 2013])

Parallelization of Density Matrix Renormalization Group (DMRG) algorithm



(Figure source: Markus Reiher)

- Distribute load of k number of sites on each node
- Perform computations in a tree structure