Scalable Tensor Algorithms for Modern Computing Systems

Suraj KUMAR

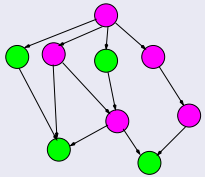
Inria ROMA Applicant

June 6, 2021

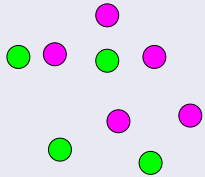
Backup Slides

Our Performance Bound

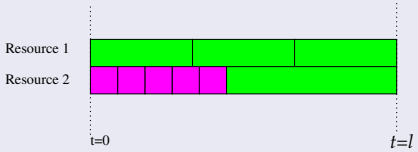
Task Graph



No Dependencies

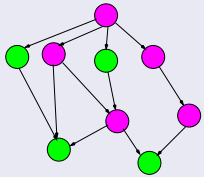


Minimum execution time (minimize l)

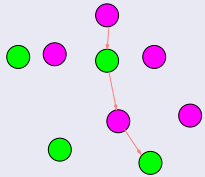


Our Performance Bound

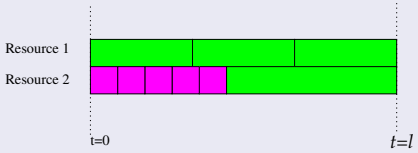
Task Graph



Some Dependencies



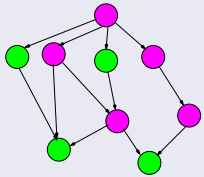
Minimum execution time (minimize l)



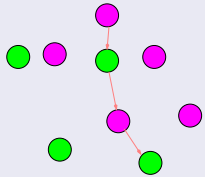
- ⋆ If any path in the graph is larger than l

Our Performance Bound

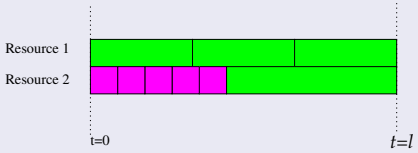
Task Graph



Some Dependencies



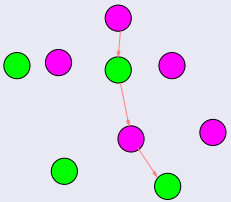
Minimum execution time (minimize l)



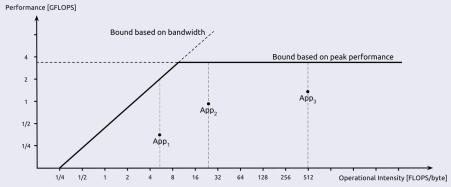
- ⋆ If any path in the graph is larger than l
  - add this path as a constraint and repeat the procedure

Roofline model with Dependencies

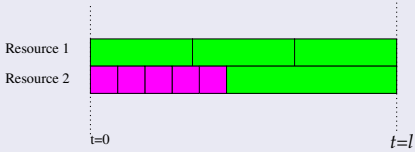
Our bound



Roofline Model



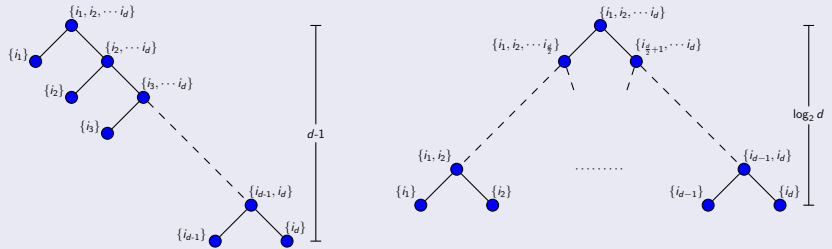
Minimum execution time (minimize l)



- ⋆ 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

Unfolding matrices of a tensor



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

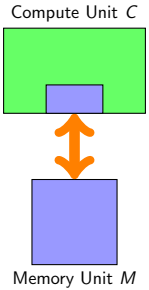
- Size of A_k is $(\prod_{l=1}^{k} n_l) \times (\prod_{l=k+1}^{d} n_l)$
- $r_k = \text{rank}(A_k)$
- Each node works with an unfolding matrix

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

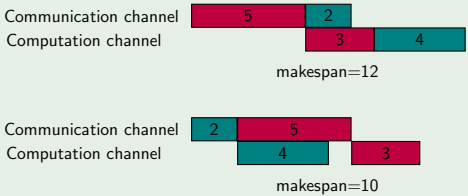
Communication Computation Overlap

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

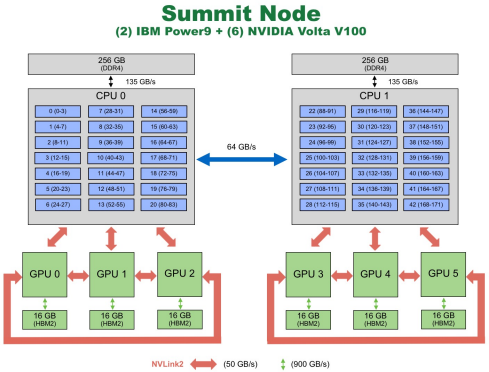
- First sort compute intensive tasks in increasing order of their data transfer time
- Then sort the communication intensive tasks in decreasing order of their computation time

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)



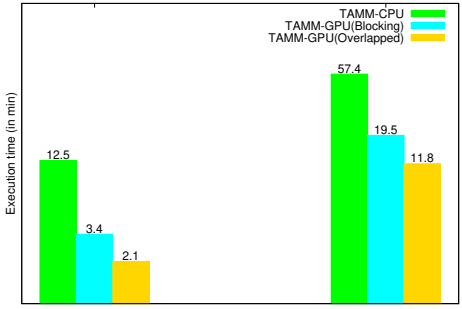
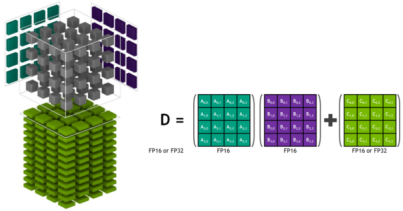
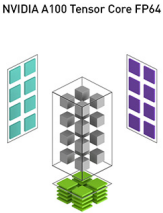


Fig source: `https://www.olcf.ornl.gov`

Architecture Aware Algorithms





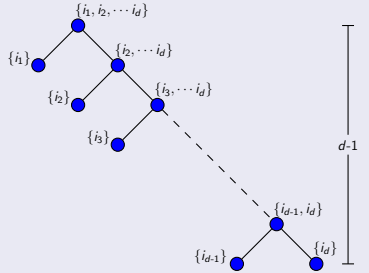
- Recent Nvidia GPUs have tensor cores to accelerate AI computations
- Most linear algebra computations do not take advantages of these units
- Design algorithms which take architecture details into account

Fig source: `www.nvidia.com`

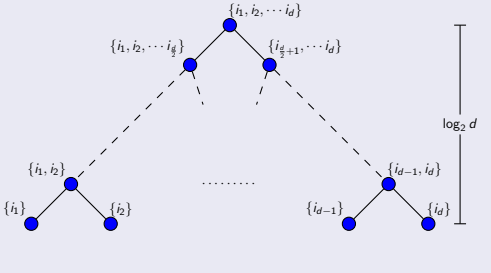
Order of separation of dimensions in tensor train algorithms

- Determine the order of separation of dimensions

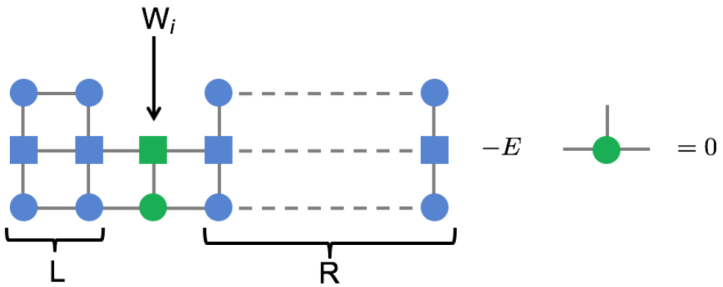
Sequential algorithm



Our parallel algorithm



Parallelization of Density Matrix Renormalization Group (DMRG) algorithm



(Fig source: Markus Reiher)

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