Parallel tensor contractions

PHY905-004 project

Fei Yuan

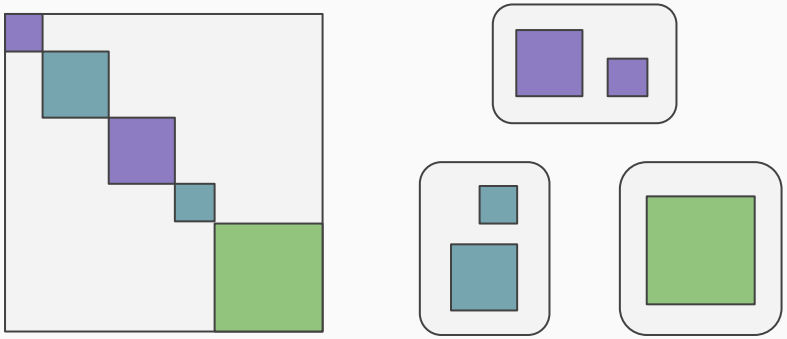
Motivation

- Tensor contractions arise naturally in various many-body theories such as CC, IM-SRG, MBPT, …
- Bulk of the computational cost in these methods: efficient tensor contractions ⇒ efficient overall performance

$$T_{pqrs} = \sum_{a \in A} \sum_{i \in I} A_{pais} B_{iqra}$$

Storage of tensors

- Tensors can take up a lot of space
- Naive storage is impractical
- Store the "diagonal" blocks only
- Split them evenly over all nodes

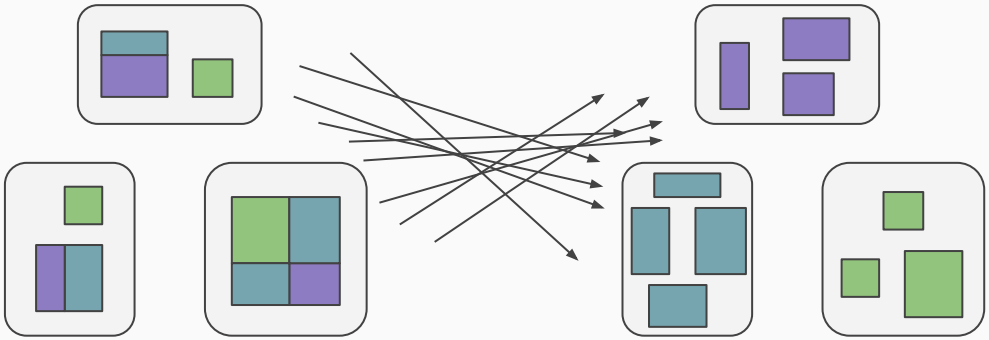


$$T_{p,q,r,s} = \delta_{(L_p+L_q),(L_r+L_s)} t^{(L_p+L_q)}_{p,q,r,s}$$

Contractions ≅ matrix multiplication

To leverage existing optimized libraries, we want to map the contraction into matrix multiplication.

This requires transposing the tensor indices: $$T'_{ps;rq} = T_{pqrs}$$



Focus of this project

How do we store the data?

How do we perform the transposition efficiently?

Can we keep memory usage among the nodes as even as possible?

Load balancing algorithm

- Keep memory usage even among nodes.
- Reduce the amount of unnecessary off-node data movement.

Initial distribution of blocks: use the simple *LPT algorithm*

Transposition: use a tweaked LPT algorithm where we assign blocks to the nodes so as to maximize the local (inter-node) data movement

Distribution algorithm is performed on the root node

Data movement

Once the distribution is known, each node will communicate with every other node to redistribute the blocks.

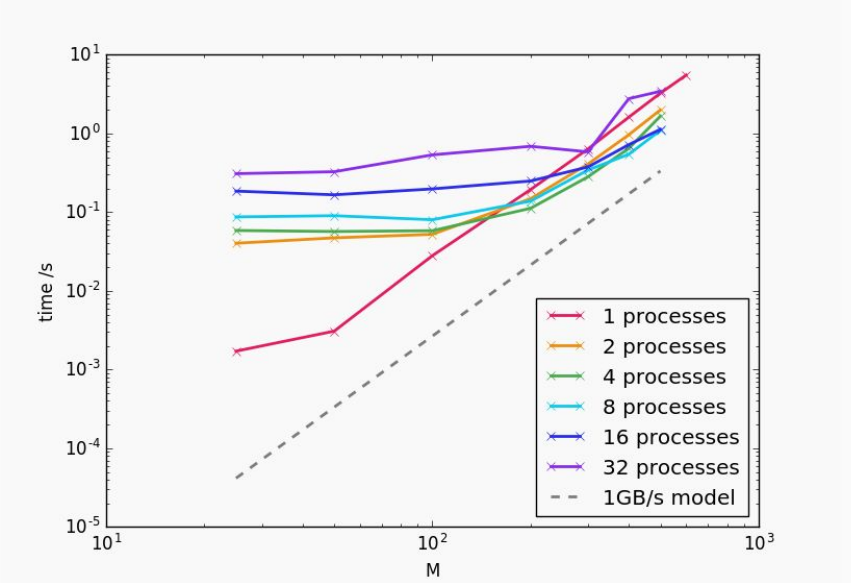
Essentially: *MPI_Isend + MPI_Irecv + MPI_Waitall*

Messages are packed using *MPI_Pack* to avoid sending a message for every block fragment.

Benchmarks

- Initialize matrix on every node with random blocks of sizes 0^2, 1^2, ..., M^2
- Emulate transpose by randomly deciding which block each element goes

- Model: (M^3/3)*8/(1GB/s)



Future goal: dynamic load balancing

- We want to keep memory usage and computation cost even among the nodes,
- But: memory ~ O(N^2), cost ~ O(N^3) (matrix multiplication).
- Can we load balance dynamically as the computation goes? (i.e. work stealing algorithm)