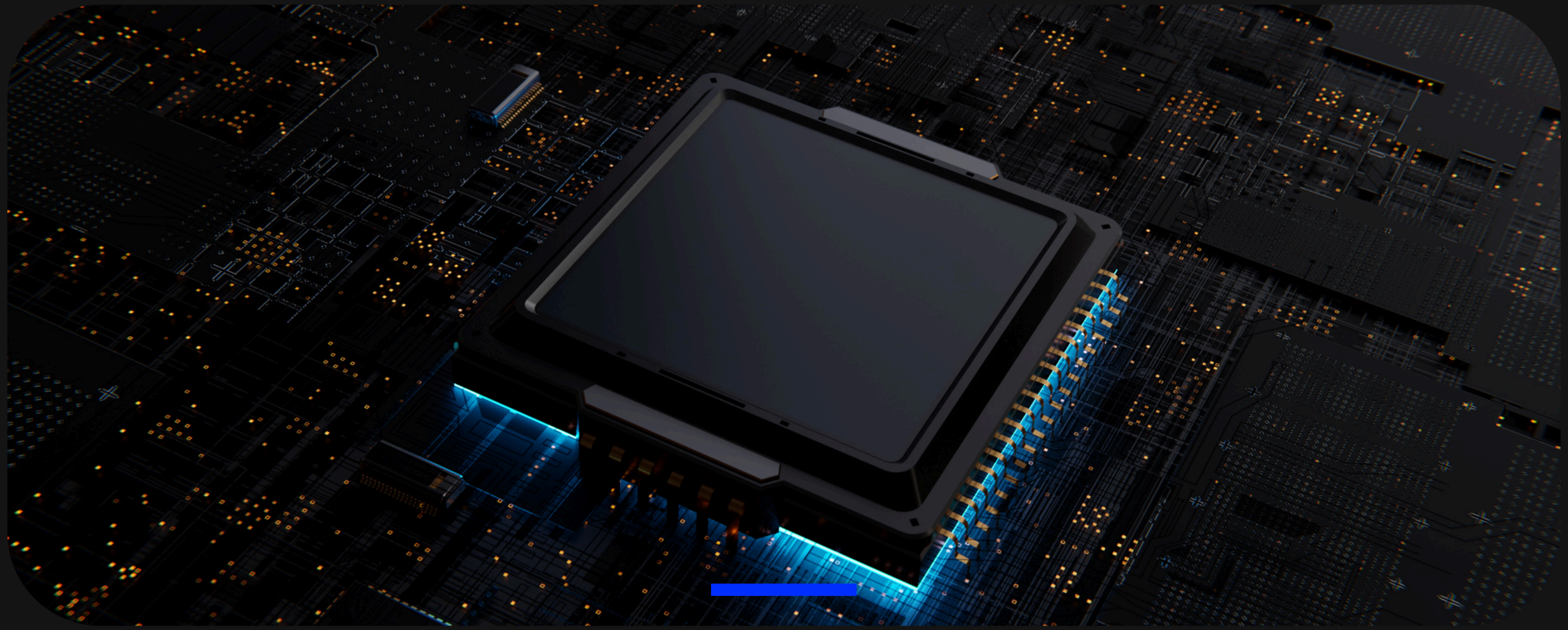
A Community Detection-Based Parallel Algorithm for Quantum Circuit Simulation



Presented By:
Umer Farooq
Irtaza Khan
Hussain Waseem

Why Simulate Quantum Circuits Classically ?

- Current quantum hardware (NISQ era) is noisy and limited.
- Classical simulations validate quantum algorithms and benchmark hardware.

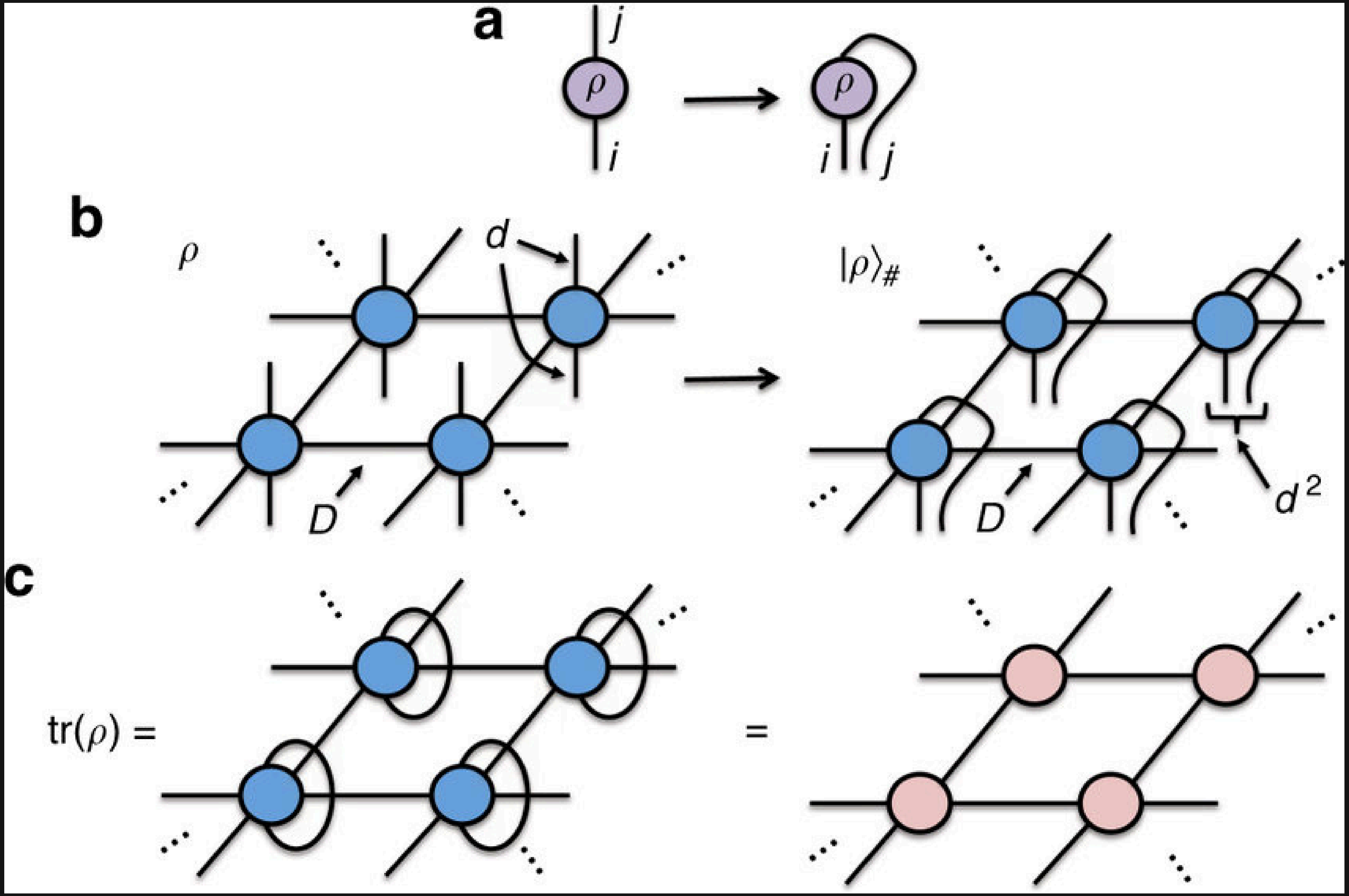
- **Challenges**:
  - Exponential memory growth with qubits (e.g., 50 qubits ≈ 9 PB).
  - High computational cost of tensor network contraction.

Tensor Networks & Quantum Circuits

Tensor Networks as a Solution

- Represent quantum circuits as graphs:
  - Nodes = quantum gates (tensors)
  - Edges = qubits (indices)
- **Contraction :** Merge tensors pairwise to compute amplitudes
- **Cost :** Order of contraction impacts time/memory (NP-hard problem)

Diagram of a Tensor Network



FEATURES AFFECTING PERFORMANCE

Feature	Benefits in Tensor Networks
Graph Partitioning	Smarter grouping of Contraction
Edge Minimization	Smaller shared states (qubits)
Ordering optimization	Faster Simulation, Less Memory

Existing Parallelization Strategies

Using Slicing :

- Slice indices to split the network → Contract sub-networks in parallel.
- High communication overhead.

GPU-Based Contraction :

- Parallelize tensor-pair contractions on GPU (e.g., cuTENSOR)
- Limited by GPU memory and tensor ranks.

Limitation :

- Spatial cost dominates for large circuits.

Key Innovation

Balances spatial and temporal costs via community-level parallelism.

Proposed 3-Staged Algorithm

Community Detection

Partition tensor network into weakly connected communities.

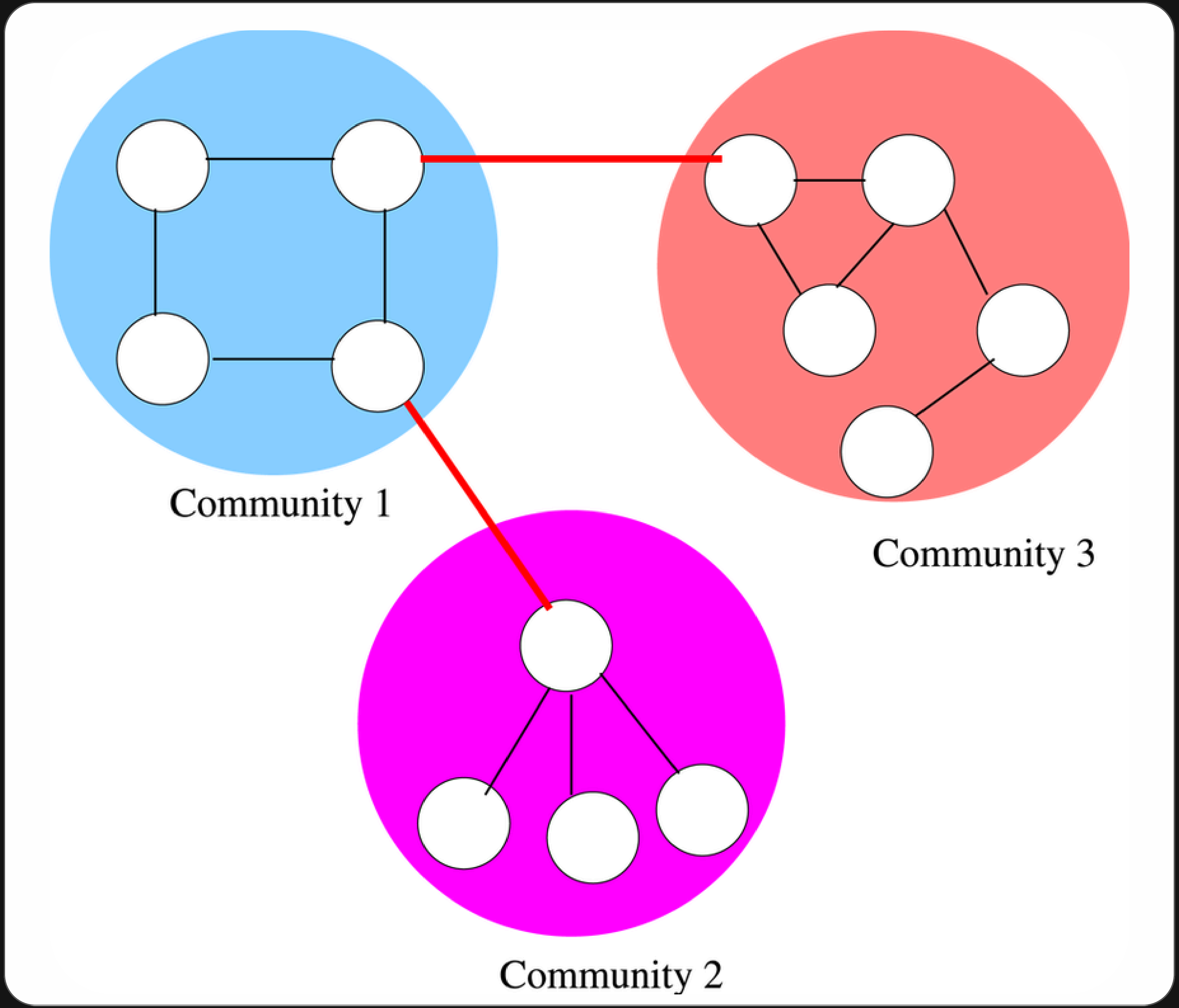
Parallel Community Contraction

Contract communities in parallel.

Final Network Contraction

Sequentially contract the reduced network.

Girvan–Newman Algorithm



Steps :

- Calculate edge betweenness centrality
- Remove edges with highest centrality → Split into communities
- Repeat until desired partitions

Facilitates in :

- Community Detection
- Community Contraction

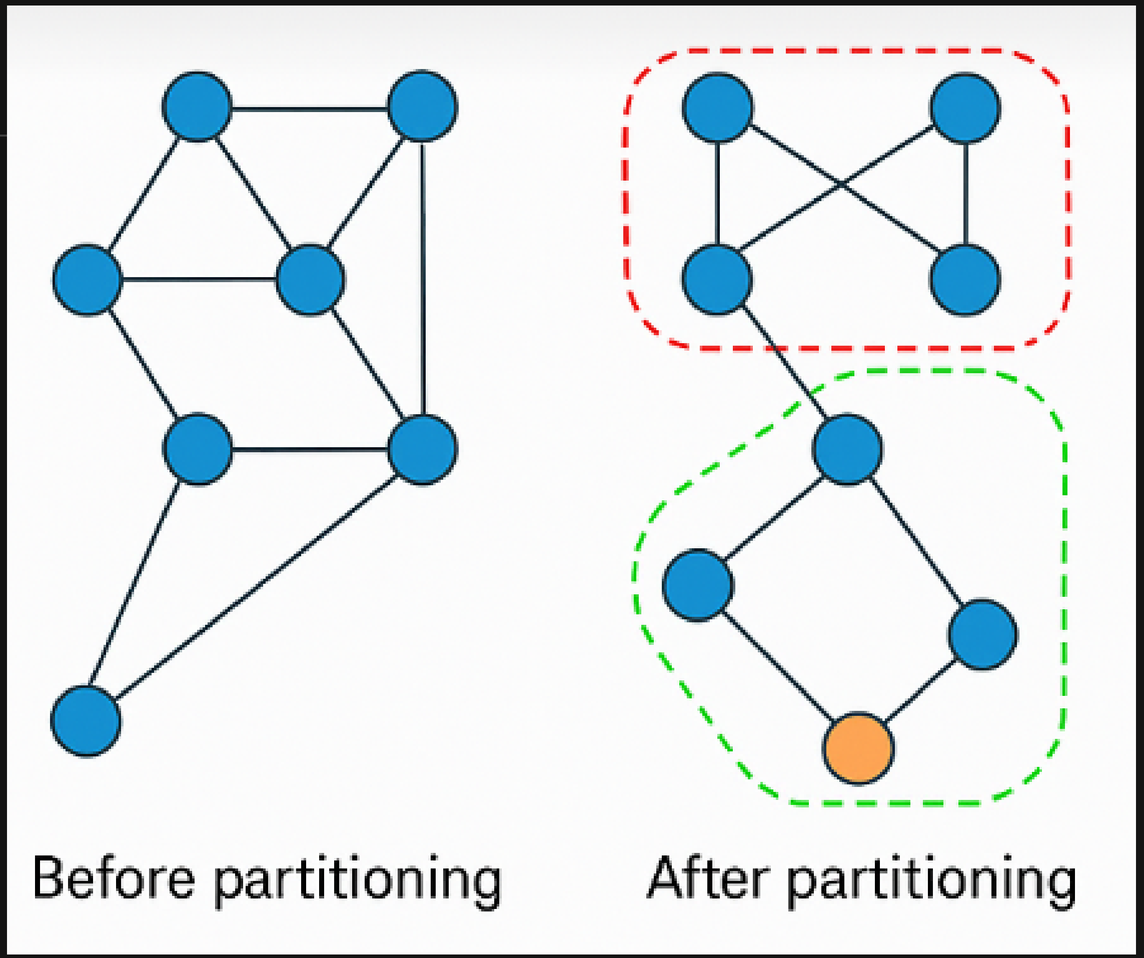
Proposed Parallelization Strategy

- Partition tensor graph with **METIS**
- Distribute communities with **MPI**
- Contract with **OpenMP/OpenCL**

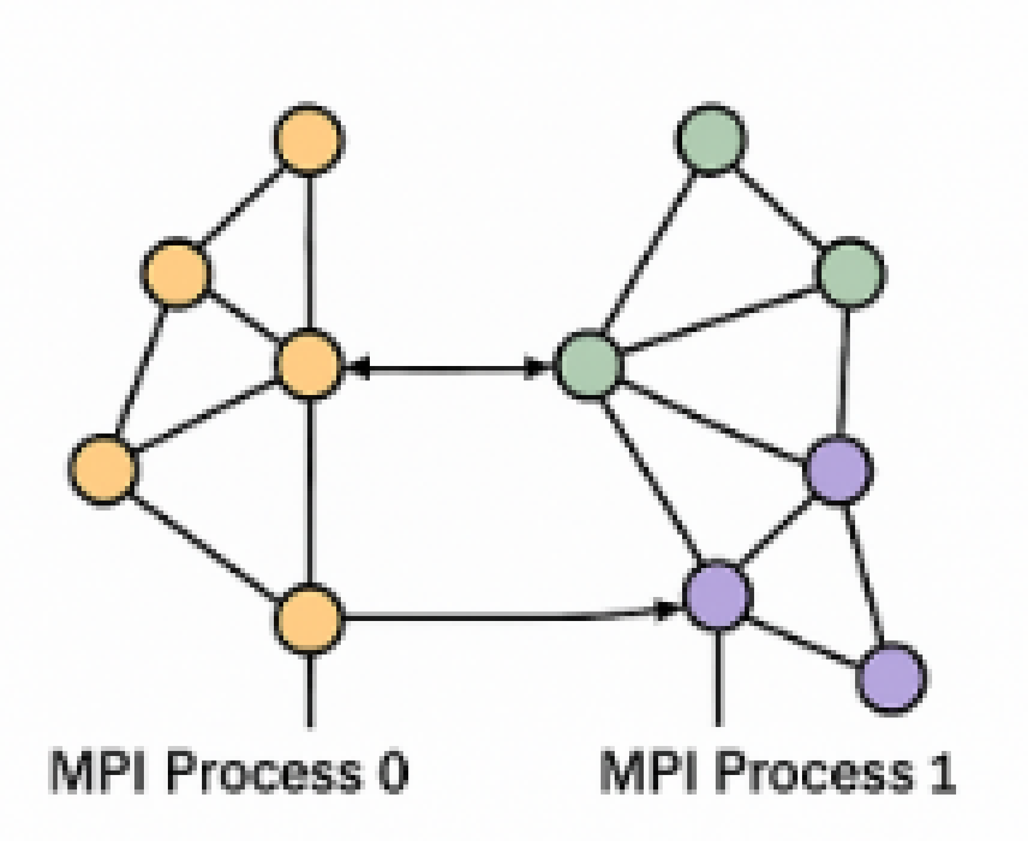
What is METIS ?

A Software Package designed for:

- Efficient Graph Partitioning
- Compute of Fill-reducing Orderings of sparse matrix
- Minimizing the number of edges cut across partitions



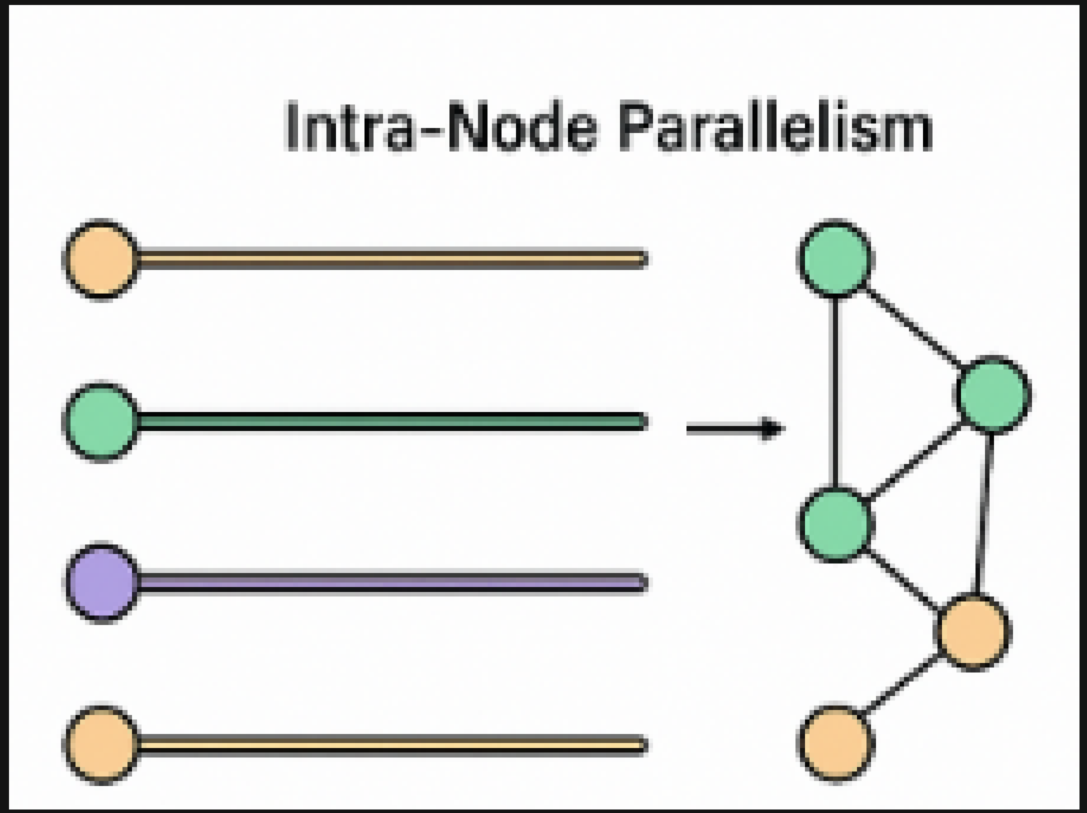
MPI for Inter-Node Parallelism



- Each METIS-partitioned subgraph is assigned to a different MPI process.
- Each process contracts its own subgraph independently.
- Once local contractions are done, inter-process edges are contracted with minimal coordination.

OpenMP/OpenCL for Intra-Node Parallelism

- Inside each MPI process:
  - Use OpenMP to parallelize tensor pair contractions across CPU cores.
  - Use OpenCL to offload contractions to GPU cores for even faster execution.



Challenges & Mitigation

Challenges

- Load imbalance
- MPI communication overhead
- GPU memory limits

Mitigation

- METIS balancing constraints
- Non-blocking MPI
- Optimized OpenCL kernels

Conclusion

- Parallelism on all levels: node-level, process-level, and graph-level.
- Better load balancing via METIS vs Girvan-Newman.
- Scalable to large systems and high-precision simulations.

ANY QUESTIONS?