Topology-aware Parallel Algorithms for Symmetric Tensor Contractions

Edgar Solomonik

University of California, Berkeley
Computer Science Division

SIAM PP'12

Outline

Electronic Structure Calculation via Coupled Cluster

Coupled Cluster (CC) is a computational method for solving the Schrodinger equation,

$$H|\Psi\rangle = E|\Psi\rangle,$$

In CC, the approximate wave-function is

$$|\Psi\rangle = e^{\hat{T}}|\Phi\rangle$$

where $|\Phi\rangle$ is the Slater determinant. The $\hat{T}$ operator in CC has the form

$$\hat{T} = \hat{T}_1 + \hat{T}_2 + \hat{T}_3 \ldots$$

where T_n is a nth rank (dimension) tensor representing nth level electron excitations.

Coupled Cluster excitation level

Computing T_n involves iteratively solving a set of non-linear equations of tensors up to dimension n. CCSD ($n = 4$) accounts for all single (S) and double (D) electron excitations. A sample contraction computed in CCSD is

$$C^{c<d}_{a<b} = \sum_{ij} A^{i<j}_{a<b} \cdot B^{c<d}_{i<j}$$

It is of interest to compute CCSDT (triples) and CCSDTQ (quadruples), which operate on tensors up to dimension 6 and 8, respectively.

Tensor contractions

Coupled Cluster motivates tensor contraction algorithms which

- **exploit symmetry in tensors**
- efficiently support contractions among tensors of diverse dimension and shape
- are suitable for long and repeated contraction sequences

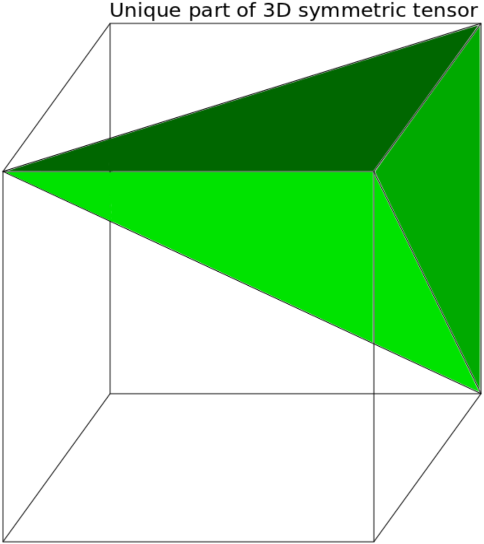
Tensor symmetry inherent to physics is computationally important

- **d-dimensional symmetry** requires a factor of **d**! **less memory**
- exploiting symmetry can also lower the arithmetic cost

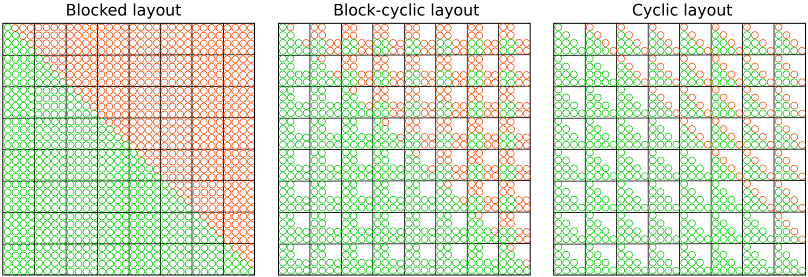
Tensor contraction design hierarchy

1. define sequence of contractions to be computed
   - done by Tensor Contraction Engine (TCE) inside NWChem
2. decompose symmetric contractions into triangular contractions
   - done by TCE inside NWChem
3. perform triangular tensor contractions
   - done with support of Global Arrays in NWChem
   - *function of Cyclops Tensor Framework*

Tensor symmetry



Blocked vs block-cyclic vs cyclic decompositions



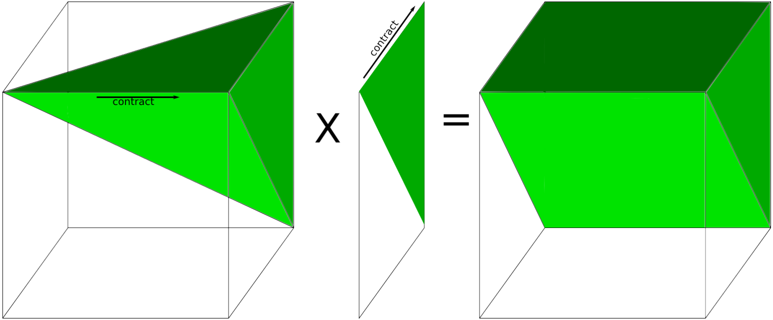
Green denotes fill (unique values) Red denotes padding / load imbalance

Cyclops Tensor Framework (CTF)

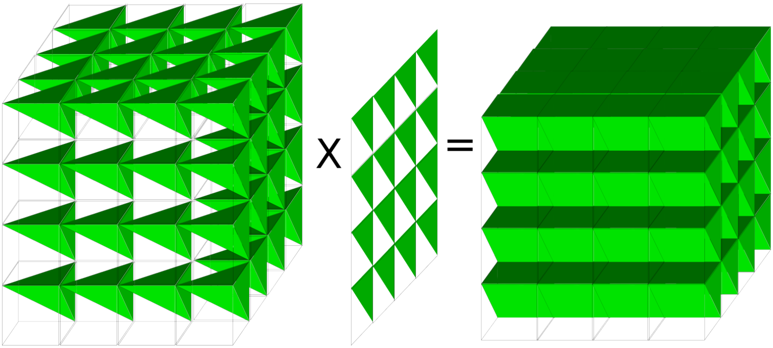
Big idea:

- decompose tensors cyclically among processors
- pick cyclic phase to preserve partial/full symmetry

3D tensor contraction

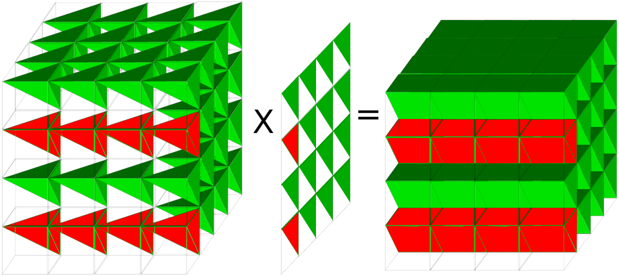


3D tensor cyclic decomposition



3D tensor mapping

Red portion denotes what processor (2,1) owns



P_{11}	P_{12}	P_{13}	P_{14}
P_{21}	P_{22}	P_{23}	P_{24}

A cyclic layout is still challenging

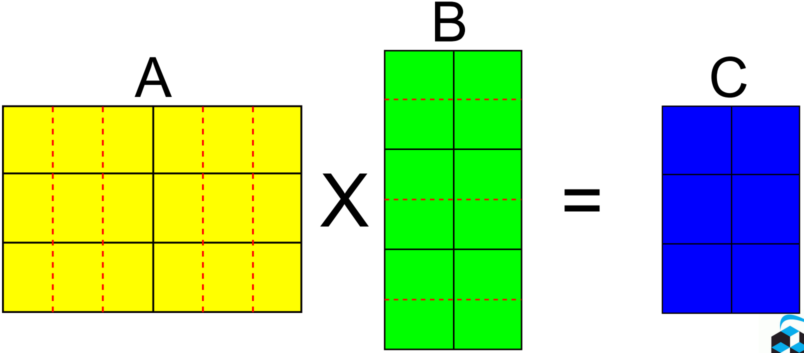
- In order to retain partial symmetry, all symmetric dimensions of a tensor must be mapped with the same cyclic phase
- The contracted dimensions of A and B must be mapped with the same phase
- And yet the virtual mapping, needs to be mapped to a physical topology, which can be any shape

Virtual processor grid dimensions

- Our virtual cyclic topology is somewhat restrictive and the physical topology is very restricted
- Virtual processor grid dimensions serve as a new level of indirection
    - If a tensor dimension must have a certain cyclic phase, adjust physical mapping by creating a virtual processor dimension
    - Allows physical processor grid to be 'stretchable'

Virtual processor grid construction

Matrix multiply on 2x3 processor grid. Red lines represent virtualized part of processor grid. Elements assigned to blocks by cyclic phase.



Cyclops Tensor Framework (CTF)

Input:

- Tensors (dimension, edge lengths, symmetries)
- Tensor data (written by global index)
- Indexed operation (contraction, summation, scale)
- Sequential kernel

Contraction algorithm:

1. Search for best tensor mapping satisfying operational constraints
2. Redistribute tensors accordingly
3. Run distributed contraction using sequential kernel

Output:

- Read tensor data by global index

CTF status

General framework

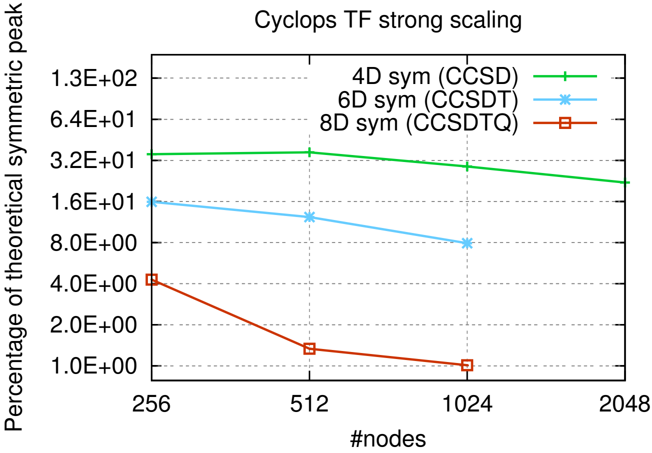
- supports tensors with any dimension, symmetry, or shape
- performs any 1,2,3 tensor operation given sequential kernel
- maps onto a torus network of any dimension and shape

Development status

- mostly functional but not fully tested
- lacking good symmetric sequential contraction kernels
- performance for 'DGEMM' contractions, e.g.

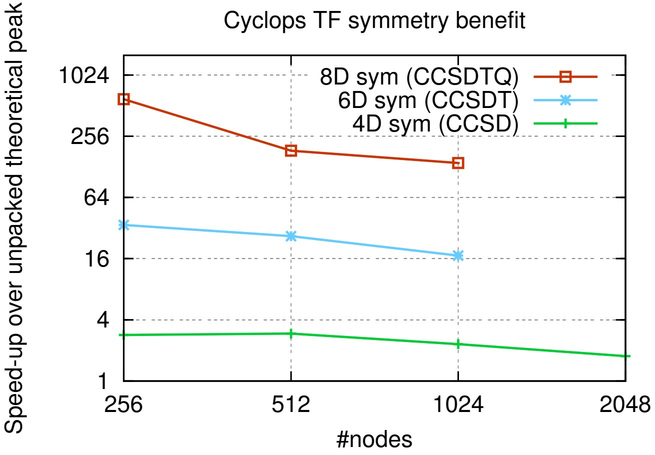
$$C^{c<d}_{a<b} = \sum_{ij} A^{i<j}_{a<b} \cdot B^{c<d}_{i<j}$$

Preliminary performance results on Blue Gene/P

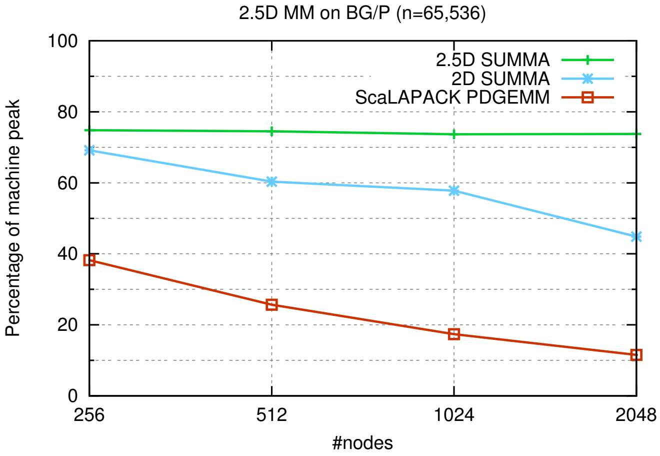


Benefit from using symmetry (preliminary)

Memory savings (CCSD 4x), (CCSDT 36x), (CCSDTQ 576x)



Performance target: 2.5D algorithms



Ongoing and future work

Testing and verification

- combinatorial explosion of symmetries and processor grids with dimension
- robust verification of low-dimensional contractions
- testing of contractions specific to Coupled Cluster

Performance optimizations

- memory-aware computing: 2.5D algorithms
- better performance models and mapping heuristics
- tuning on BG/Q architecture

New features

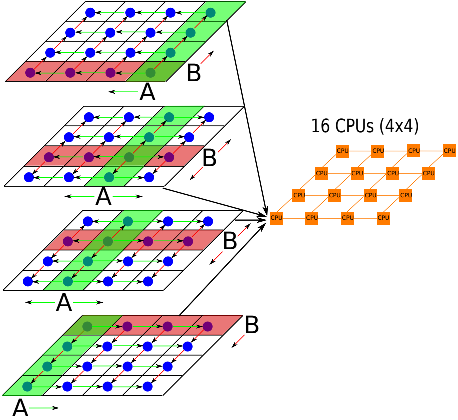
- efficient sequential symmetric contraction kernels
- tensor sparsity

Acknowledgments

- Collaborators
  - Jeff Hammond (ALCF, ANL)
  - Devin Matthews (UT Austin)
  - James Demmel (UC Berkeley)
- Krell CSGF DOE fellowship
- Resources at Argonne National Lab and Lawrence Berkeley National Lab
- Berkeley ParLab funding
  - Microsoft (Award #024263) and Intel (Award #024894)
  - U.C. Discovery (Award #DIG07-10227)

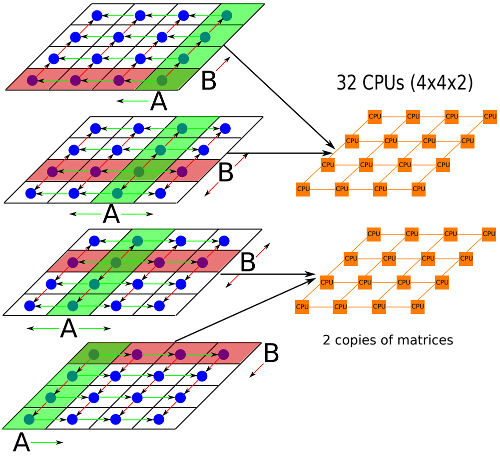
Backup slides

2D matrix multiplication

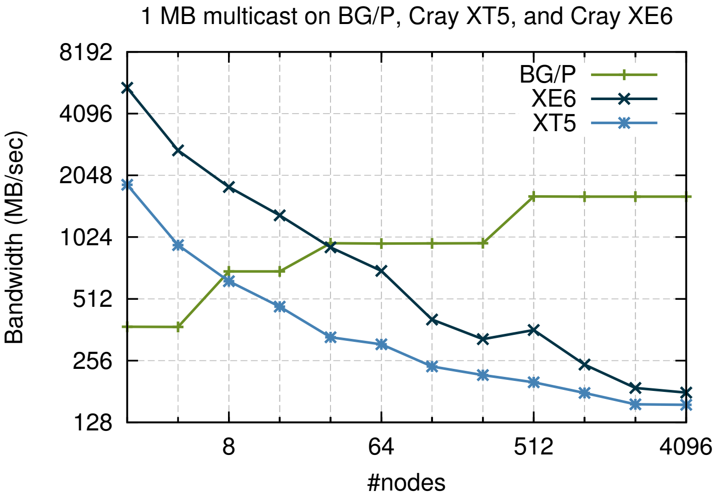


[Cannon 69], [Van De Geijn and Watts 97]

2.5D matrix multiplication



Performance of multicast (BG/P vs Cray)



Preliminary performance results on Hopper (Cray XE6)