A massively-parallel framework for Coupled Cluster

Edgar Solomonik (in collaboration with Devin Matthews, Jeff Hammond, and James Demmel)
solomon@cs.berkeley.edu

Tensor contractions

Aim: support parallel execution of tensor contractions such as those needed by CCD,

$$R_{ij}^{ab} = V_{ij}^{ab} + P(ia,jb)\Big[T_{ij}^{ae} I_e^b - T_{im}^{ab} I_j^m + \frac{1}{2} V_{ef}^{ab} T_{ij}^{ef} + \frac{1}{2} T_{mn}^{ab} I_{ij}^{mn} - T_{mj}^{ae} I_{ie}^{mb} - I_{ie}^{ma} T_{mj}^{eb} + (2T_{mi}^{ea} - T_{im}^{ea}) I_{ej}^{mb}\Big]$$

$$\begin{aligned} I_b^a &= (-2V_{eb}^{mn} + V_{be}^{mn}) T_{mn}^{ea} \\ I_j^i &= (2V_{ef}^{mi} - V_{ef}^{im}) T_{mj}^{ef} \\ I_{kl}^{ij} &= V_{kl}^{ij} + V_{ef}^{ij} T_{kl}^{ef} \\ I_{jb}^{ia} &= V_{jb}^{ia} - \frac{1}{2} V_{eb}^{im} T_{jm}^{ea} \\ I_{bj}^{ia} &= V_{bj}^{ia} + V_{be}^{im} (T_{mj}^{ea} - \frac{1}{2} T_{mj}^{ae}) - \frac{1}{2} V_{be}^{mi} T_{mj}^{ae} \end{aligned}$$

would like to have extensiblity to CCSDT/CCSDTQ methods with 6-dimensional and 8-dimensional symmetric tensors.

Cyclops Tensor Framework (CTF)

Cyclops Tensor Framework is a parallel C++ framework which provides support for symmetric tensors

```
CTF_World(MPI_COMM_WORLD) w;
CTF_Matrix(n_o, n_o, SY, w) I;
CTF_Tensor(4, {n_o, n_o, n_v, n_v}, {AS, NS, AS, NS}, w) T;
CTF_Tensor(4, {n_o, n_o, n_v, n_v}, {AS, NS, AS, NS}, w) R;
```

arbitrary contractions are supported via index strings

```
R["abij"] += 1.0 * T["aeij"] * I["be"];
```

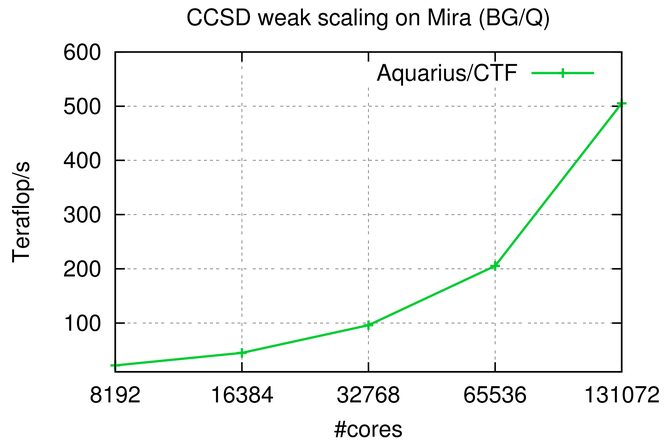
Each contraction is executed in a massively-parallel fashion. Data may be entered and read by global index

```
for(i = 0; i < len; i++)
  data[i] = foo(indices[i]));
A.write_remote_data(len, indices, data);
```

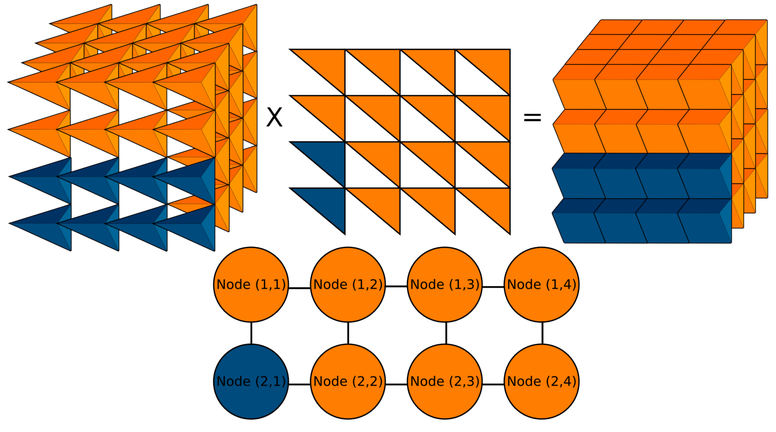
This disributed functionality layer is employed by higher-level chemistry libraries such as Aquarius and QChem which provide spin-symmetric tensor types and implement Coupled Cluster.

CCSD weak scaling

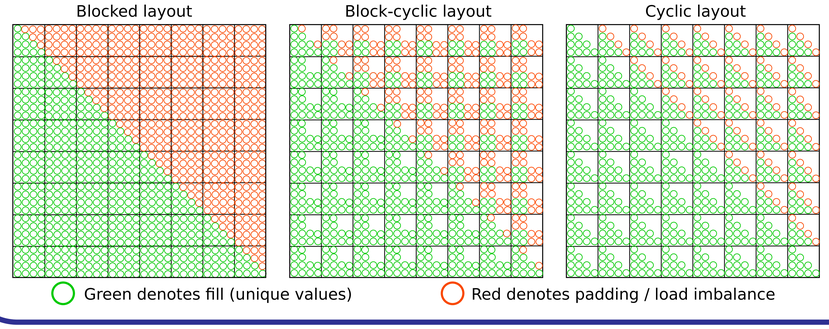
Cyclops Tensor Framework scales CCSD to $n_o = 250, n_v = 1,000$.



Virtualized topology-aware mapping



Cyclic data-layout



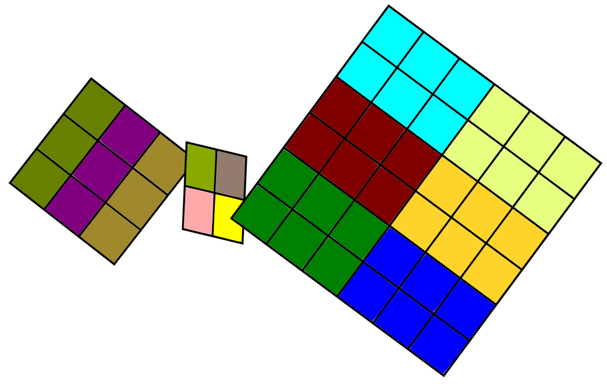
CCSD comparison with NWChem

CCSD iteration time on 64 nodes of Cray XE6 (Hopper), compares favorably to NWChem:

system	n_o	n_v	CTF	NWChem
w5	25	180	14 sec	36 sec
w7	35	252	90 sec	178 sec
w9	45	324	127 sec	-
w12	60	432	336 sec	-

On 128 nodes, CTF completed w9 in 73 sec/iter, NWChem in 223 sec/iter.

Nested parallel matrix multiplication



When possible, tensor data is also replicated to reduce communication.

Tensor transposition

Three data reshuffling stages are needed, kernels for each are threaded

1. Sparse writes input tensor data
2. Tensor globally redistributed to map up each contraction
3. Tensor blocks partially unpacked and locally tansposed

Future work

- Sparse tensor support
- Improved algorithms for broken tensor symmetries
- CCSDT (currently in optimization) CCSDTQ (to be implemented)
- Tensor slicing (e.g. `B = A["1 : n_o/2, 1 : n_o/2, 1 : n_v, 1 : n_v"];`)