Leveraging modern supercomputing infrastructure for tensor contractions in large electronic-structure calculations

Ilya A. Kaliman

University of Southern California

September 18-19, 2017
Tensors in Quantum Chemistry

\[ \hat{H} \psi = E \psi \]

**Coupled Cluster Equations**

\[
D_{ij}^{ab} = \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b
\]

\[
T_{ij}^{ab} D_{ij}^{ab} = \langle ij||ab \rangle + \mathcal{P}_-(ab) \left( \sum_c f_{bc} t_{ij}^{ac} - \frac{1}{2} \sum_{kld} \langle kl||cd \rangle t_{kl}^{bd} t_{ij}^{ac} \right)
\]

\[
- \mathcal{P}_-(ij) \left( \sum_k f_{jk} t_{ik}^{ab} + \frac{1}{2} \sum_{kld} \langle kl||cd \rangle t_{ij}^{cd} t_{ik}^{ab} \right)
\]

\[
+ \frac{1}{2} \sum_{kl} \langle ij||kl \rangle t_{kl}^{ab} + \frac{1}{4} \sum_{kld} \langle kl||cd \rangle t_{ij}^{cd} t_{kl}^{ab} + \frac{1}{2} \sum_{cd} \langle ab||cd \rangle t_{ij}^{cd}
\]

\[
- \mathcal{P}_-(ij) \mathcal{P}_-(ab) \left( \sum_{kc} \langle kb||jc \rangle t_{ik}^{ac} - \frac{1}{2} \sum_{kld} \langle kl||cd \rangle t_{ij}^{db} t_{ik}^{ac} \right)
\]
Tensors in Quantum Chemistry

- Tensors of floating point numbers are used extensively in high-level electronic-structure calculations
- 4-index tensors are common Coupled Cluster methods
- Contractions are the most expensive step
- Complex structure of tensors – must use symmetry and sparsity
- Huge data size (many terabytes)
- Large calculations can take weeks
Q-Chem Quantum Chemistry Package

Q-Chem

ccman2 – Coupled Cluster module

libcc – library of CC equations

libtensor (frontend)

Native backend

libxm backend

CTF backend

This work
Data storage using block tensors

Permutational symmetry
\[ a_{ji} = -a_{ij} \]

Spin symmetry

Molecular point-group symmetry

**Canonical tensor blocks**

**Non-canonical blocks (computed from canonical blocks)**

**Zero blocks**
Block tensor operations

Contractions

\[
\begin{align*}
C_{11} & = A_{11} \otimes B_{11} + A_{21} \otimes B_{12} \\
C_{12} & = A_{12} \otimes B_{11} + A_{22} \otimes B_{12}
\end{align*}
\]

Unfolding + BLAS/BLIS

Additions

\[
\begin{align*}
C_{11} & = A_{11} \otimes B_{11} + A_{21} \otimes B_{12} \\
C_{12} & = A_{12} \otimes B_{11} + A_{22} \otimes B_{12}
\end{align*}
\]

- Only non-zero canonical blocks (orange) need to be computed
- Blocks can be computed independently in parallel
Calculations on a single node

Shared Memory

Canonical tensor blocks
Calculations on a supercomputer

Can this scale?
Calculations on a supercomputer

Can this scale? It can! (with a fast cache)
BurstBuffer on NERSC Cori

6.5 Gb/sec read/write bandwidth

http://www.nersc.gov/users/computational-systems/cori/burst-buffer/burst-buffer/
Implementation and benchmarks: libxm

- Libxm is a library of primitive tensor operations
  - `xm_contract(1.0, A, B, 2.0, C, “abcd”, “ijcd”, “ijab”);`
  - `xm_add(1.0, A, 2.0, B, “ij”, “ji”);`
  - ...

- Main components
  - MPI-aware disk-backed memory allocator
  - Code for tensor operations
  - Auxiliary routines

- Stores all data on disk

- Hybrid MPI/OpenMP parallel design
  - Static load balancing between the nodes (MPI)
  - Dynamic load balancing within a node (OpenMP)

- https://github.com/ilyak/libxm
Libxm parallel scaling on Cori

<table>
<thead>
<tr>
<th>Nodes</th>
<th>xm_contract</th>
<th>xm_add</th>
<th>xm_set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (32 cores)</td>
<td>23660 (1.0x)</td>
<td>787 (1.0x)</td>
<td>457 (1.0x)</td>
</tr>
<tr>
<td>2 (64 cores)</td>
<td>11771 (2.0x)</td>
<td>436 (1.8x)</td>
<td>324 (1.4x)</td>
</tr>
<tr>
<td>4 (128 cores)</td>
<td>5938 (4.0x)</td>
<td>203 (3.9x)</td>
<td>115 (4.0x)</td>
</tr>
<tr>
<td>8 (256 cores)</td>
<td>3167 (7.5x)</td>
<td>168 (4.7x)</td>
<td>66 (6.9x)</td>
</tr>
<tr>
<td>16 (512 cores)</td>
<td>1606 (14.7x)</td>
<td>69 (11.4x)</td>
<td>28 (16.3x)</td>
</tr>
<tr>
<td>32 (1024 cores)</td>
<td>836 (28.3x)</td>
<td>32 (24.6x)</td>
<td>21 (21.8x)</td>
</tr>
</tbody>
</table>

Total tensor data size is over 2 Tb, time in seconds, speedup relative to one node in parenthesis
Conclusions

- A new distributed-parallel model for tensor operations is implemented in the *libxm* library
- Shared filesystem is used as an inter-node common storage for tensors
- Data size is not limited by the amount of RAM or number of nodes
- The hybrid MPI/OpenMP parallel code shows excellent scaling when adequate data caching is employed
Thank you!

• Acknowledgments
  – Prof. Anna Krylov, USC
  – Dr. Evgeny Epifanovskiy, Q-Chem

https://github.com/ilyak/libxm