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

$$\hat{H}\,\psi = E\,\psi$$

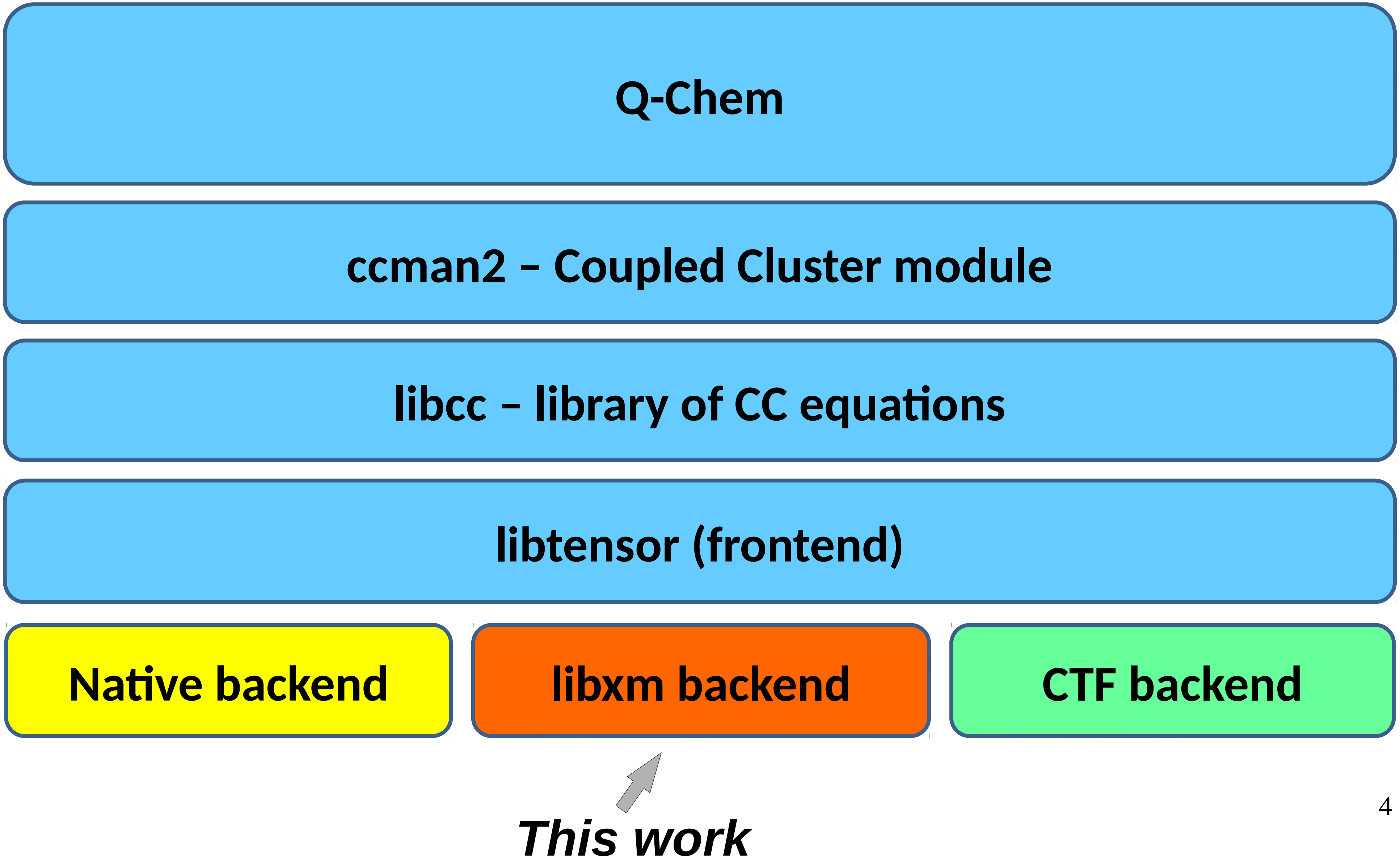
Coupled Cluster Equations

$$
\begin{aligned}
D_{ij}^{ab} &= \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b \\
T_{ij}^{ab} D_{ij}^{ab} &= \langle ij||ab\rangle + \mathcal{P}_-(ab)\left(\sum_c f_{bc} t_{ij}^{ac} - \frac{1}{2}\sum_{klcd}\langle kl||cd\rangle t_{kl}^{bd} t_{ij}^{ac}\right) \\
&- \mathcal{P}_-(ij)\left(\sum_k f_{jk} t_{ik}^{ab} + \frac{1}{2}\sum_{klcd}\langle kl||cd\rangle t_{jl}^{cd} t_{ik}^{ab}\right) \\
&+ \frac{1}{2}\sum_{kl}\langle ij||kl\rangle t_{kl}^{ab} + \frac{1}{4}\sum_{klcd}\langle kl||cd\rangle t_{ij}^{cd} t_{kl}^{ab} + \frac{1}{2}\sum_{cd}\langle ab||cd\rangle t_{ij}^{cd} \\
&- \mathcal{P}_-(ij)\mathcal{P}_-(ab)\left(\sum_{kc}\langle kb||jc\rangle t_{ik}^{ac} - \frac{1}{2}\sum_{klcd}\langle kl||cd\rangle t_{lj}^{db} t_{ik}^{ac}\right)
\end{aligned}
$$

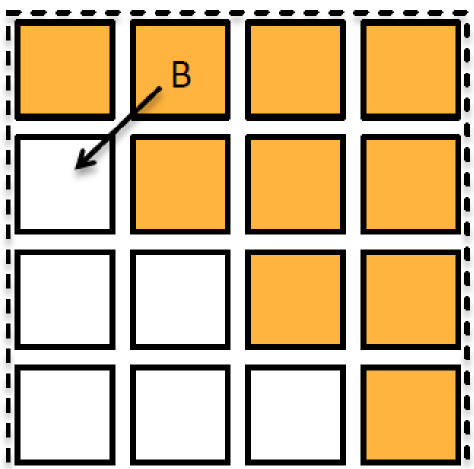
Tensors in Quantum Chemistry

- 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

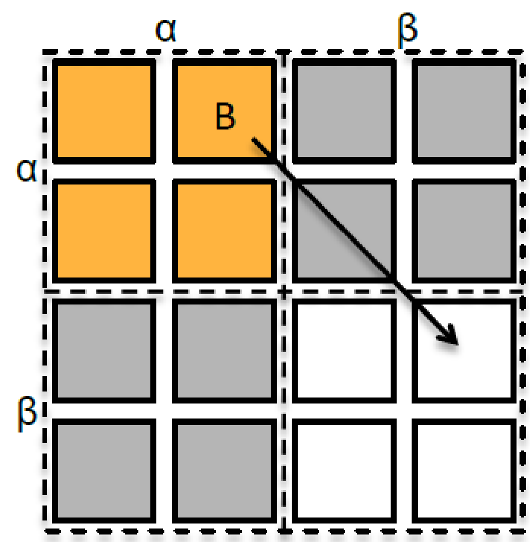
Q-Chem Quantum Chemistry Package



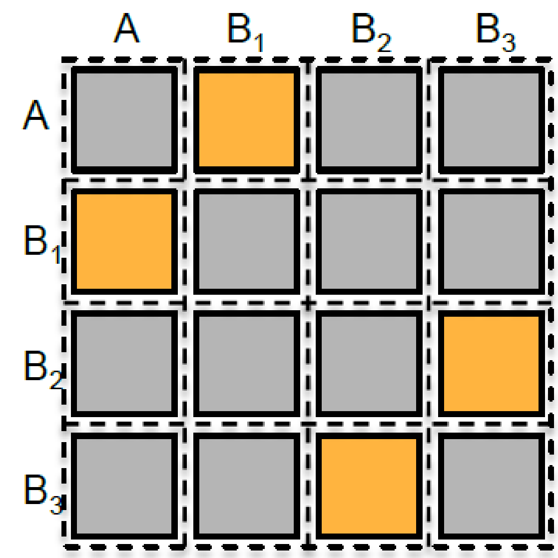
Data storage using block tensors



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

$$
\begin{bmatrix} C_{11} & C_{21} \\ C_{12} & C_{22} \\ C_{13} & C_{23} \end{bmatrix}
=
\begin{bmatrix} A_{11} & A_{21} \\ A_{12} & A_{22} \\ A_{13} & A_{23} \end{bmatrix}
\times
\begin{bmatrix} B_{11} & B_{21} \\ B_{12} & B_{22} \end{bmatrix}
$$

$$C_{11} = \underbrace{A_{11} \otimes B_{11}}_{\text{Unfolding + BLAS/BLIS}} + \cancelto{0}{A_{21} \otimes B_{12}}$$

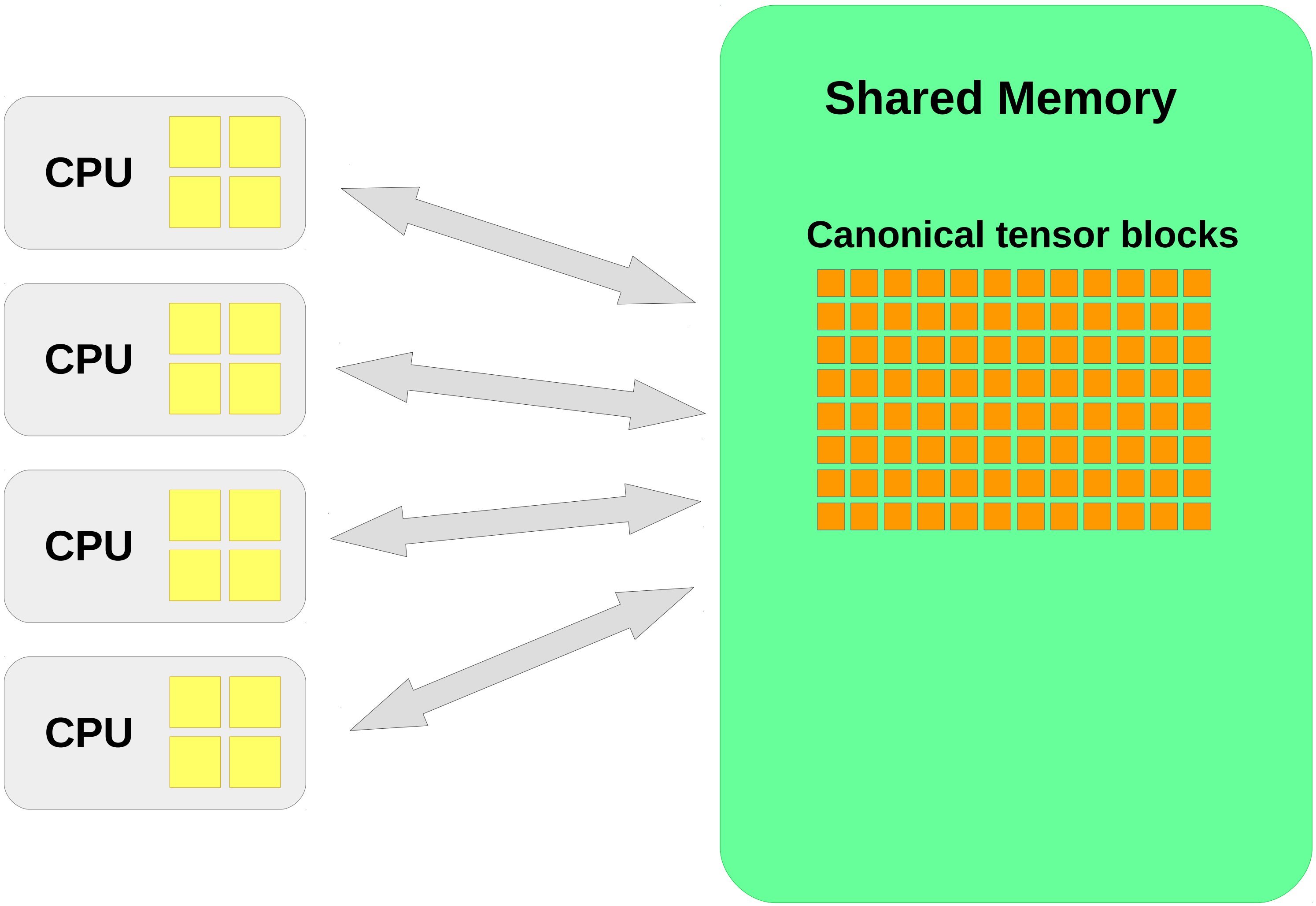
$$C_{12} = \cancelto{0}{A_{12} \otimes B_{11}} + \cancelto{0}{A_{22} \otimes B_{12}}$$

Additions

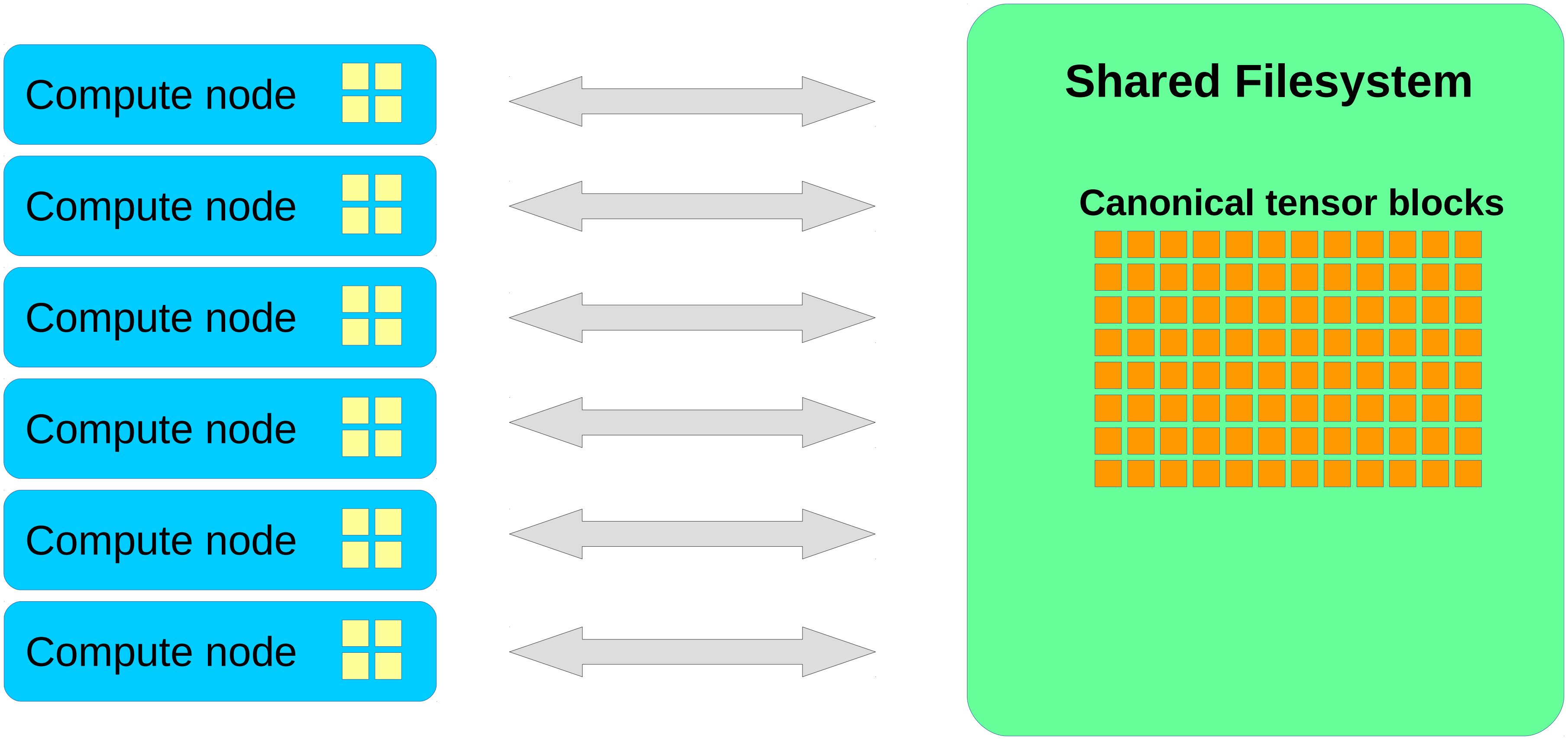
$$
\begin{bmatrix} C_{11} & C_{21} & C_{31} \\ C_{12} & C_{22} & C_{32} \\ C_{13} & C_{23} & C_{33} \end{bmatrix}
=
\begin{bmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \\ A_{13} & A_{23} & A_{33} \end{bmatrix}
+
\begin{bmatrix} B_{11} & B_{21} & B_{31} \\ B_{12} & B_{22} & B_{32} \\ B_{13} & B_{23} & B_{33} \end{bmatrix}
$$

- **Only non-zero canonical blocks (orange) need to be computed**
- **Blocks can be computed independently in parallel**

Calculations on a single node

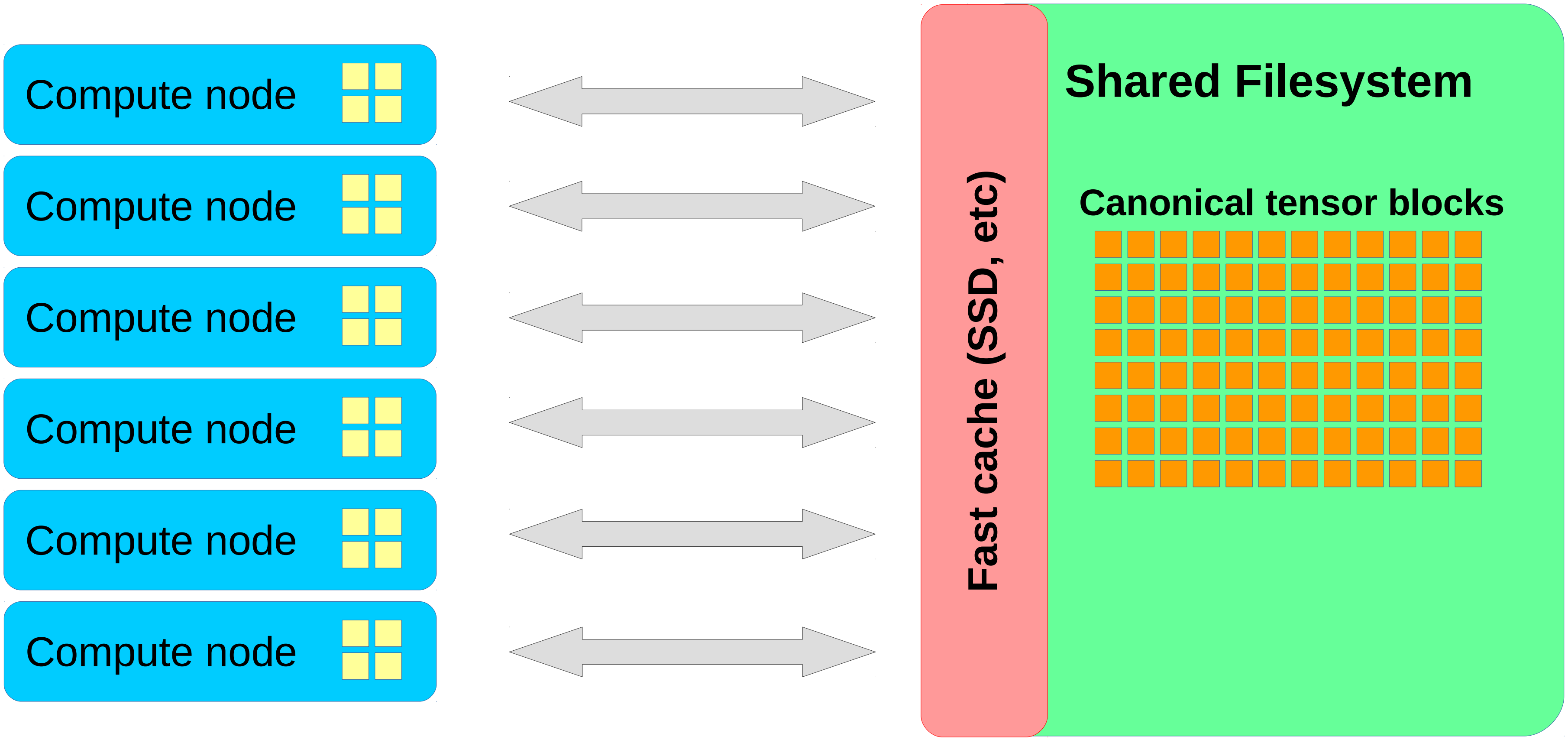


Calculations on a supercomputer



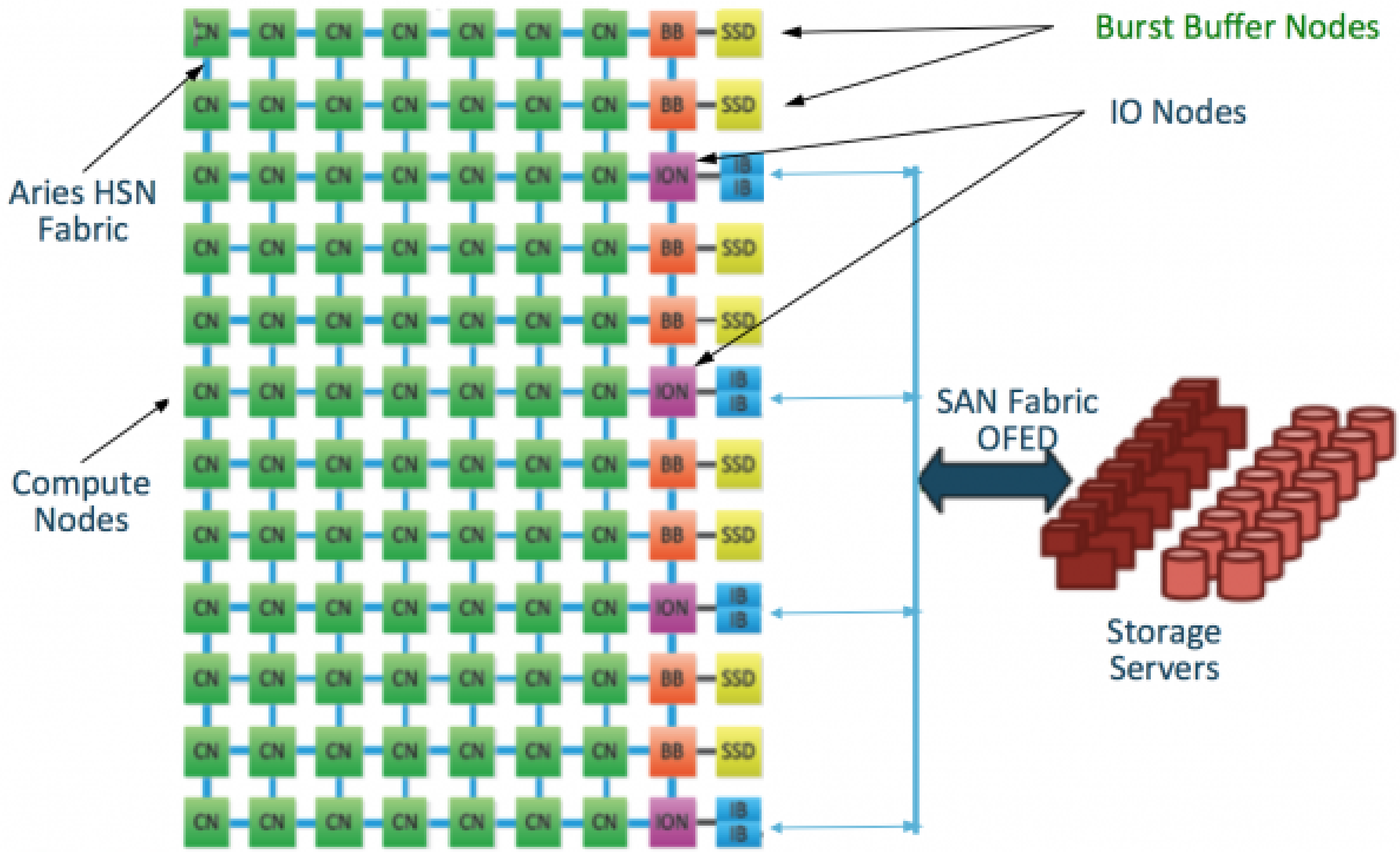
Can this scale?

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

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

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 Epifanovsky, Q-Chem

https://github.com/ilyak/libxm