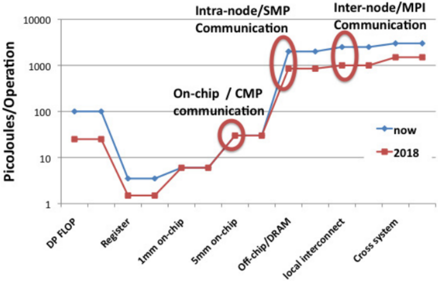
Scalable Numerical Linear Algebra for Data Science

Edgar Solomonik

University of Illinois at Urbana-Champaign

May 16, 2017

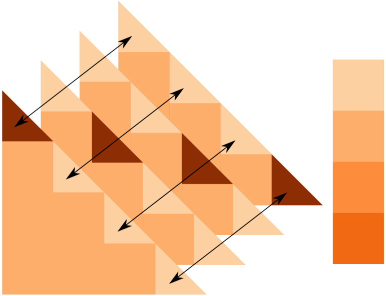
Communication-avoiding algorithms

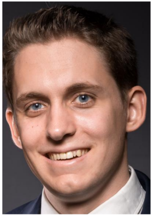


- **"Engineering FLOPs is not a design constraint – data movement presents the most daunting engineering and computer architecture challenge."** – Shalf, Dosanjh, Morrison, VECPAR 2010
- numerical computations are prevalent in all data sciences
- Goal: design fundamental numerical algorithms that achieve better scalability by avoiding data movement

Solving triangular systems of equations

Solving triangular linear system with many right-hand sides (TRSM) is a critical subroutine in most linear system solvers

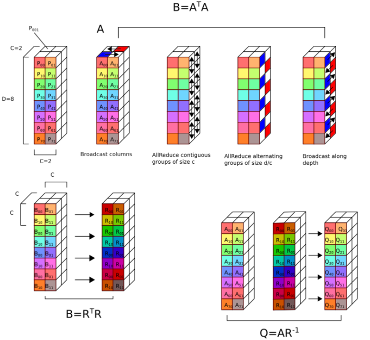




- using selective inversion of diagonal blocks decreases number of messages by $O(p^{2/3})$ on p processors with respect to known algorithms
- Bachelor thesis work by Tobias Wicky, currently working on MS thesis
- Tobias Wicky, ES, and Torsten Hoefler, Communication-avoiding parallel algorithms for solving triangular systems of linear equations, *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2017.

QR factorization (solving least-squares problems)

Extend CholeskyQR2 algorithm, obtaining ideal accuracy for well conditioned matrices ($\kappa = O(1/\sqrt{\epsilon})$), to a general parallel QR algorithm



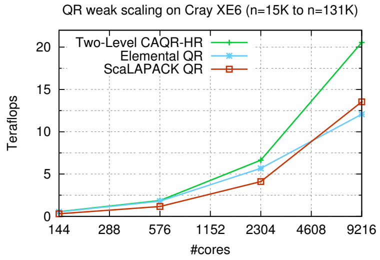
D/C simultaneous 3D Cholesky Factorizations on cubes of dimension C

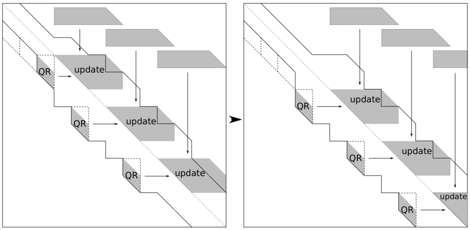
D/C simultaneous 3D Matrix Multiplications on cubes of dimension C

- new practical parallel algorithm reduces bandwidth cost by $O(p^{1/6})$ with respect to best-existing implementation
- analysis and development by Edward Hutter (BS ECE 2017, starting PhD in CS at UIUC in Fall 2017)

QR, Eigenvalue, and SVD factorizations

- QR and SVD are critical to data-fitting and compression
- new algorithms for QR factorization and eigenvalue computation for symmetric matrices faster by $O(p^{1/6})$ in communication cost





- ongoing work on SVD factorization via QR with pivoting and randomized projections

A stand-alone library for petascale tensor computations

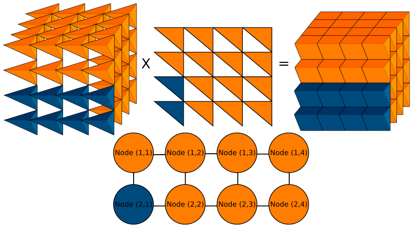
Cyclops Tensor Framework (CTF)

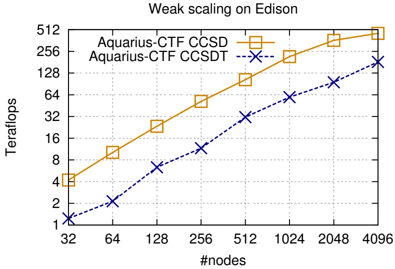
- distributed-memory symmetric/sparse tensors as C++ objects

```
Matrix<int> A(n, n, AS|SP, World(MPI_COMM_WORLD));
Tensor<float> T(order, is_sparse, dims, syms, ring, world);
T.read(...); T.write(...); T.slice(...); T.permute(...);
```

- parallel contraction/summation of tensors

```
Z["abij"] += V["ijab"];
W["mnij"] += 0.5*W["mnef"]*T["efij"];
M["ij"]   += Function<>([](double x){ return 1./x; })(v["j"]);
```





Ongoing and future work

Ongoing work and future directions in CTF

- integration with faster parallel numerical solvers
- development of new (sparse) tensor applications
  - algebraic multigrid
  - finite/spectral element methods
  - FFT, bitonic sort, parallel scan, HSS matrix computations
  - tensor factorizations and tensor networks
- existing CTF applications
  - Aquarius (lead by Devin Matthews)
  - QChem via Libtensor (lead by Evgeny Epifanovsky)
  - QBall DFT for metallic systems (lead by Eric Draeger)
  - CC4S (lead by Andreas Grüneis)
  - early collaborations involving Lattice QCD and DMRG
  - faster methods for shortest-path and graph centrality computations