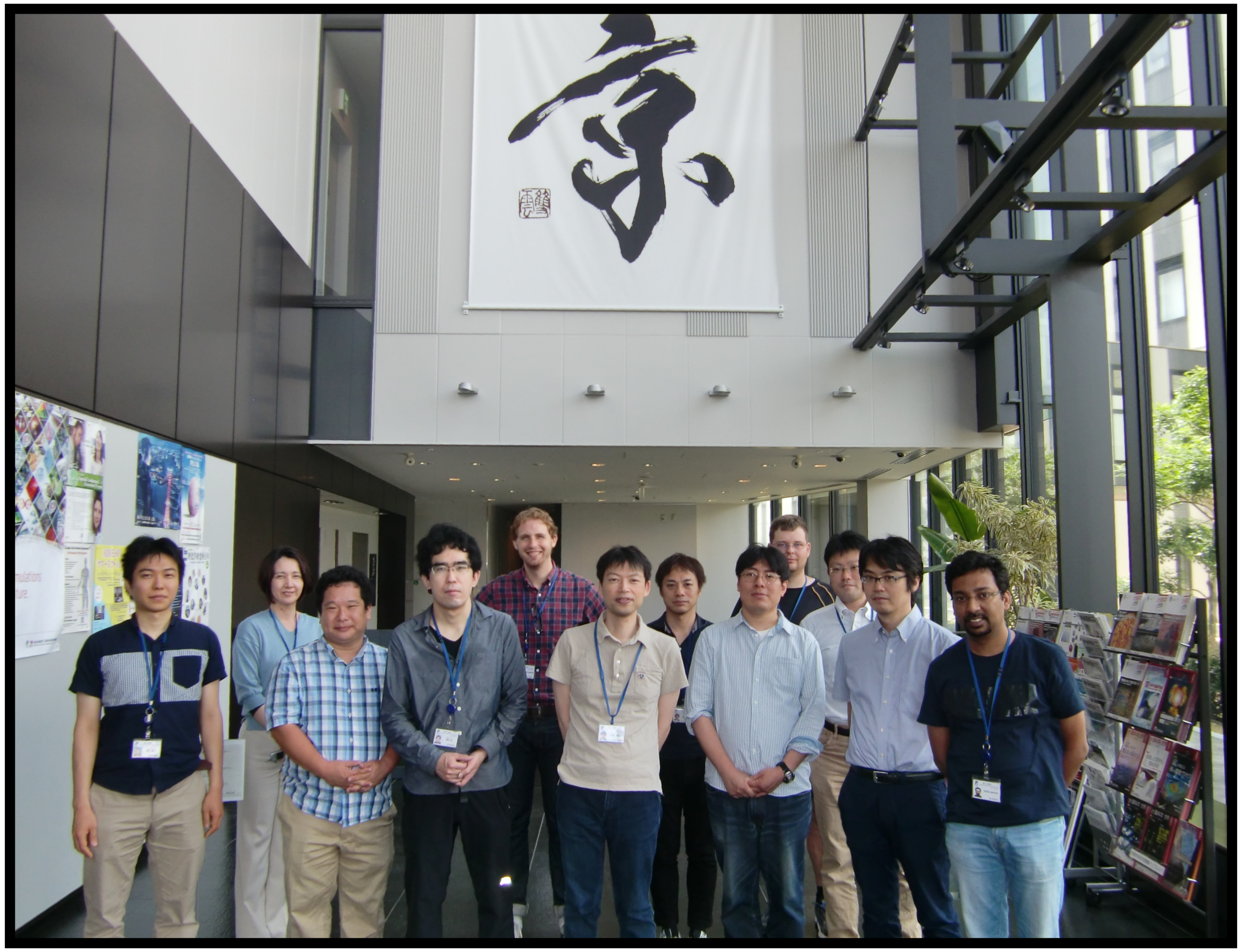
Computing Matrix Functions on the K Computer

William Dawson
Takahito Nakajima
RIKEN Center for Computational Science

2018年度第2回計算科学フォーラム

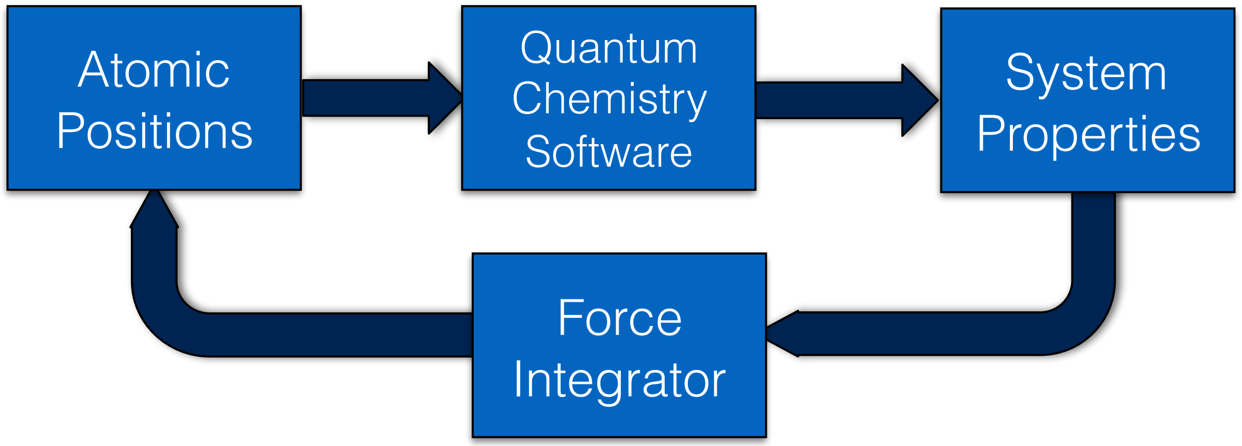
Computational Molecular Science Research Team



- Also in collaboration with Luigi Genovese (French Alternative Energies and Atomic Energy Commission), Marco Zaccaria (Boston College), Massimo Reverberi (Sapienza University of Rome).

Introduction to Computational Chemistry

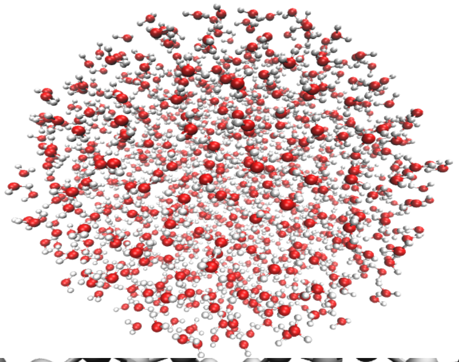
- In our team, we work to develop software and methods for understanding molecules and materials from according to the laws of quantum mechanics.

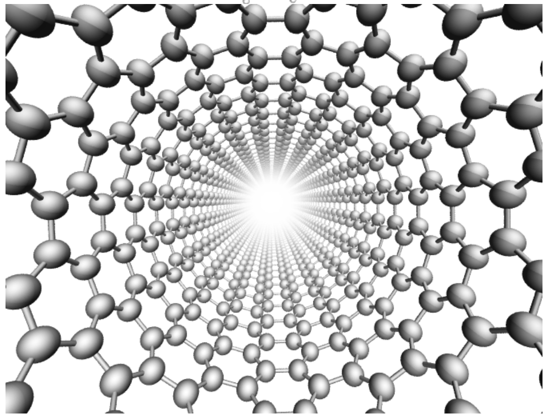


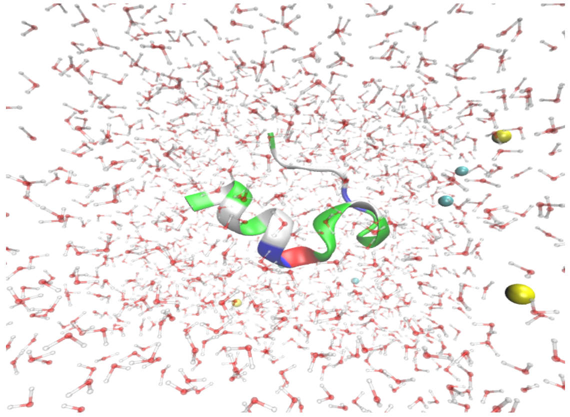
- Quantum chemistry software gives us atomic level insight, allowing us to go beyond the technical limits of experiment.
- Calculation quantities: band gaps, chemical reactions, rate constants, durability, etc.

Introduction - Large Scale Calculations

- Large uniform environment with small perturbation (e.g. dilute solutions).
- Large Nanostructures (e.g. Carbon Nanotubes).
- Molecules in a realistic environment (e.g. proteins).
- Not only perform calculations on the system, but also to gain insight into the actual chemistry.
- These calculations require clever algorithms, and large computational resources.







Ratcliff, Laura E., Stephan Mohr, Georg Huhs, Thierry Deutsch, Michel Masella, and Luigi Genovese. "Challenges in large scale quantum mechanical calculations." *Wiley Interdisciplinary Reviews: Computational Molecular Science* 7, no. 1 (2017): e1290.

Outline

- Introduction to Matrix Functions
  - Formal Definition
  - Methods of Computing Matrix Functions
  - Motivating Matrices
  - NTPoly introduction
- Parallelization Techniques
  - Distributed Memory Parallelization
  - On Node Parallelization
- Usability Considerations
  - Data Distribution
  - Programming Language Support
- Example Applications
  - Quantum Chemistry
  - Social Network Analysis
  - Search Engine Optimization

Introduction to Matrix Functions

Introduction to Matrix Functions

Cauchy integral definition:

$$f(A) := \frac{1}{2\pi i} \int_{\Gamma} f(z) (zI - A)^{-1} dz$$

where f is analytic on and inside a closed contour Γ that encloses the spectrum of A.

Simple definition: We are all familiar with functions of a single variable f(x). In the matrix function case, just replace the variable x with a matrix A.

Standard Function	**Matrix Function**	**Interpretation**
$f(x) = x^2$	$f(A) = A^2$	Matrix Product
$f(x) = 1/x$	$f(A) = A^{-1}$	Matrix Inverse
$f(x) = e^x$	$f(A) = e^A$	Matrix Differential Equation
f(x) = sign(x)	f(A) = sign(A)	Projection on to Subspace

Higham, Nicholas J. *Functions of matrices: theory and computation*. Vol. 104. Siam, 2008.

Motivating Applications

- Solving the generalized eigenvalue equation.

  $Ax = \lambda Bx \Rightarrow B^{-1/2}AB^{-1/2}x = \lambda x$ (if B is SPD).

- Constructing good preconditioners.

  $Ax = c \Rightarrow BAx = Bc$, where $B \cong A^{-1}$.

- Computing centrality measures of a network.

  A: the adjacency matrix of a graph.

  Kantz centrality: $(I - \boldsymbol{\alpha}A)^{-1}$

  Estrada Centrality: $e^{\beta A}$

- Solution to Sylvester equation, algebraic Ricatti, etc.

Motivating Applications - Chemistry

- Diagonalization free methods for quantum chemistry.
- Given the hamiltonian matrix H, we wish to construct the density matrix D.
- Usually we do this by solving the eigenvalue equation:

  $HV = \lambda V$ (where V is a n_{basis} x $n_{electrons}$ matrix).

- And compute the density matrix as $D = VV^T$.
- Instead, we can compute the density matrix directly using the fermi function:

  $D = 1/(e^{\beta(H-\mu)} + \mathbf{I})$.

Methods for Computing Matrix Functions

1. Diagonalization (if possible):

   $A = ZDZ^{-1}$.

   $f(A) = Zf(D)Z^{-1}$.

2. Schur Decomposition (explicit formulas exist for upper triangular matrices exist).

3. Taylor series expansion:

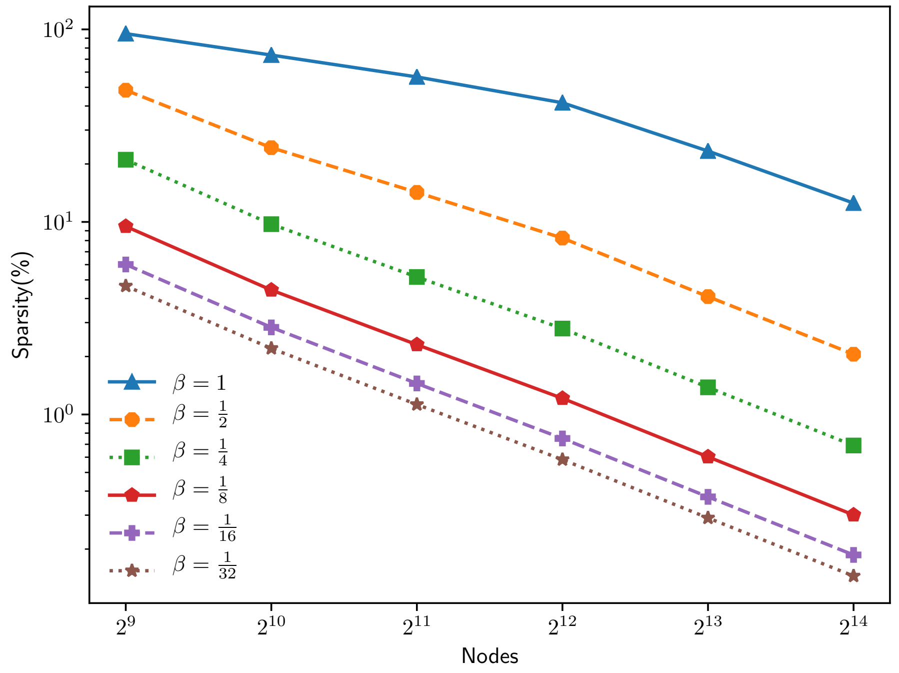
   $\cos(A) = I - A^2/2! + A^4/4! + A^6/6! \ldots$

4. Polynomial Approximation (and Rational Approximation).

5. From each function's definition (A^{-1} : $A*X = I$).

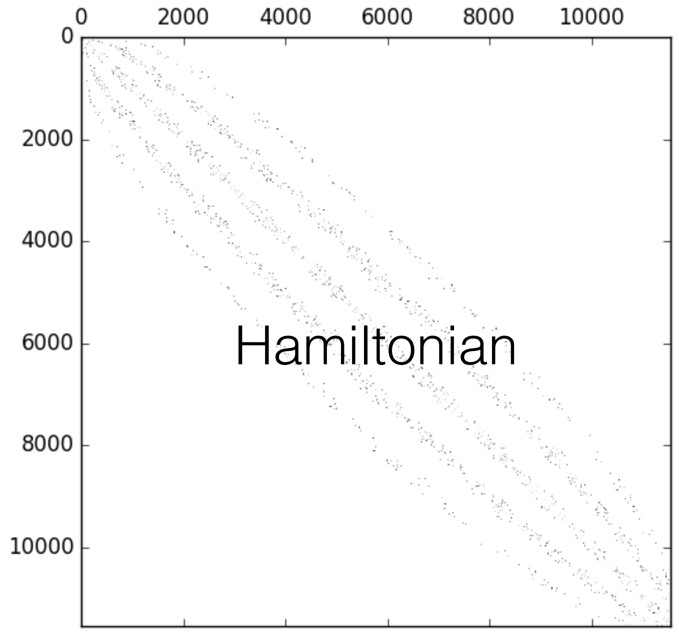
Motivating Matrices

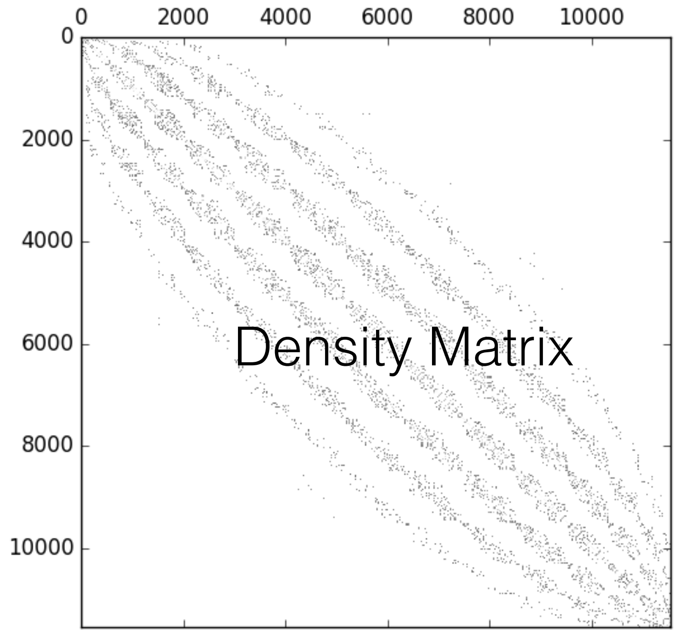
- In many domains, the problem of interest can be represented using a sparse, hermitian matrix.
- Under certain conditions, not only is the matrix A sparse, but also some matrix functions f(A) are sparse.
- Estrada matrix exponential $e^{\beta A}$ contains a scaling factor which might be interpreted as a unit of edge weight.
- For certain values of β, the matrix exponential of small world matrices is also sparse.



Motivating Matrices - Chemistry

- For insulating systems (and metals at high temperature), it is known that the density matrix is sparse when represented in a localized basis.
- Example: the Hamiltonian and Density Matrix of 1920 water molecules computed using the BigDFT code.





http://bigdft.org/

Sparsity Aware Matrix Function Calculation

- From the list of methods for computing matrix functions, we will select calculations based on matrix polynomials.

  e.g. Chebysehv polynomials: $f(A) \cong \sum c_i T_i(x)$.

  $T_0(A) = \mathbf{I}$ $\quad$ $T_2(A) = 2A^2 - \mathbf{I}$ $\quad$ $T_4(A) = 8A^4 - 8A^2 + \mathbf{I}$

  $T_1(A) = A$ $\quad$ $T_3(A) = 4A^3 - 3A$ $\quad$ …

- Computing a matrix polynomial requires only two core routines: matrix addition, matrix multiplication.
  - Easy to parallelize.
  - Many functions can be tuned through just two routines.
- In the case of sparse matrices, we replace these kernels with sparse matrix addition, and sparse matrix multiplication.

NTPoly - A Library for Computing Matrix Functions

- **General Polynomials**
  - Standard Polynomials
  - Chebyshev Polynomials
  - Hermite Polynomials
- **Transcendental Functions**
  - Trigonometric Functions
  - Exponential and Logarithm
- **Matrix Roots**
  - Square Root and Inverse Square Root
  - Matrix *p*th Root and Inverse *p*th root
- **Quantum Chemistry**
  - Density Matrix Minimization
  - Density Matrix Purification
  - Chemical Potential Calculation
  - Density Matrix Extrapolation
- **Other**
  - Matrix Inverse (and Moore-Penrose Inverse)
  - Sign Function/Polar Decomposition
  - Parallel File I/O
  - MIT License (available on Github)

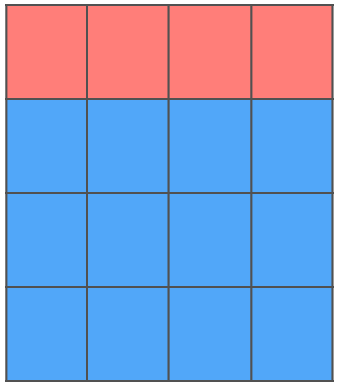
Dawson, William, and Takahito Nakajima. "Massively parallel sparse matrix function calculations with NTPoly." *Computer Physics Communications* 225 (2018): 154-165.

Parallelization Techniques

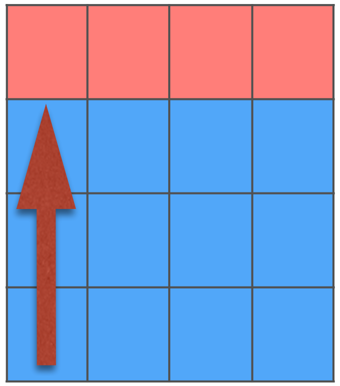
Matrix Multiplication Parallelization - 2.5D

1D Algorithm: Each processor has a matrix slice

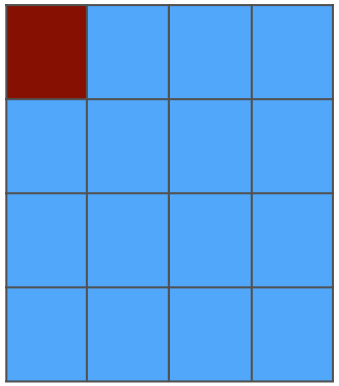
Start With A Slice



Receive Columns

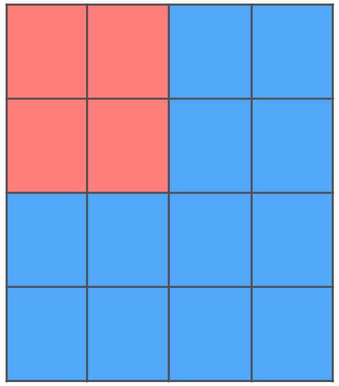


Compute Element

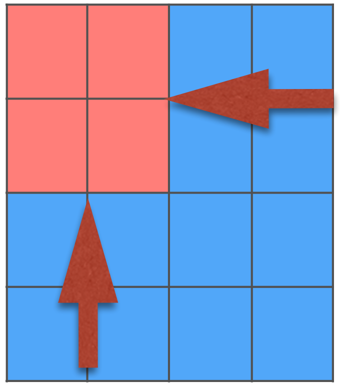


2D Algorithm: Each processor has a matrix block

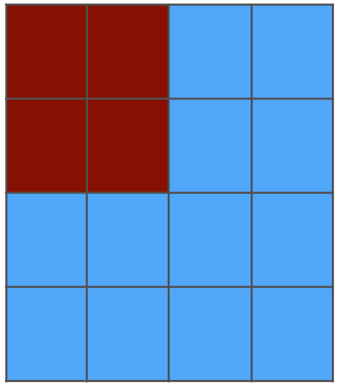
Start With A Block



Receive Blocks



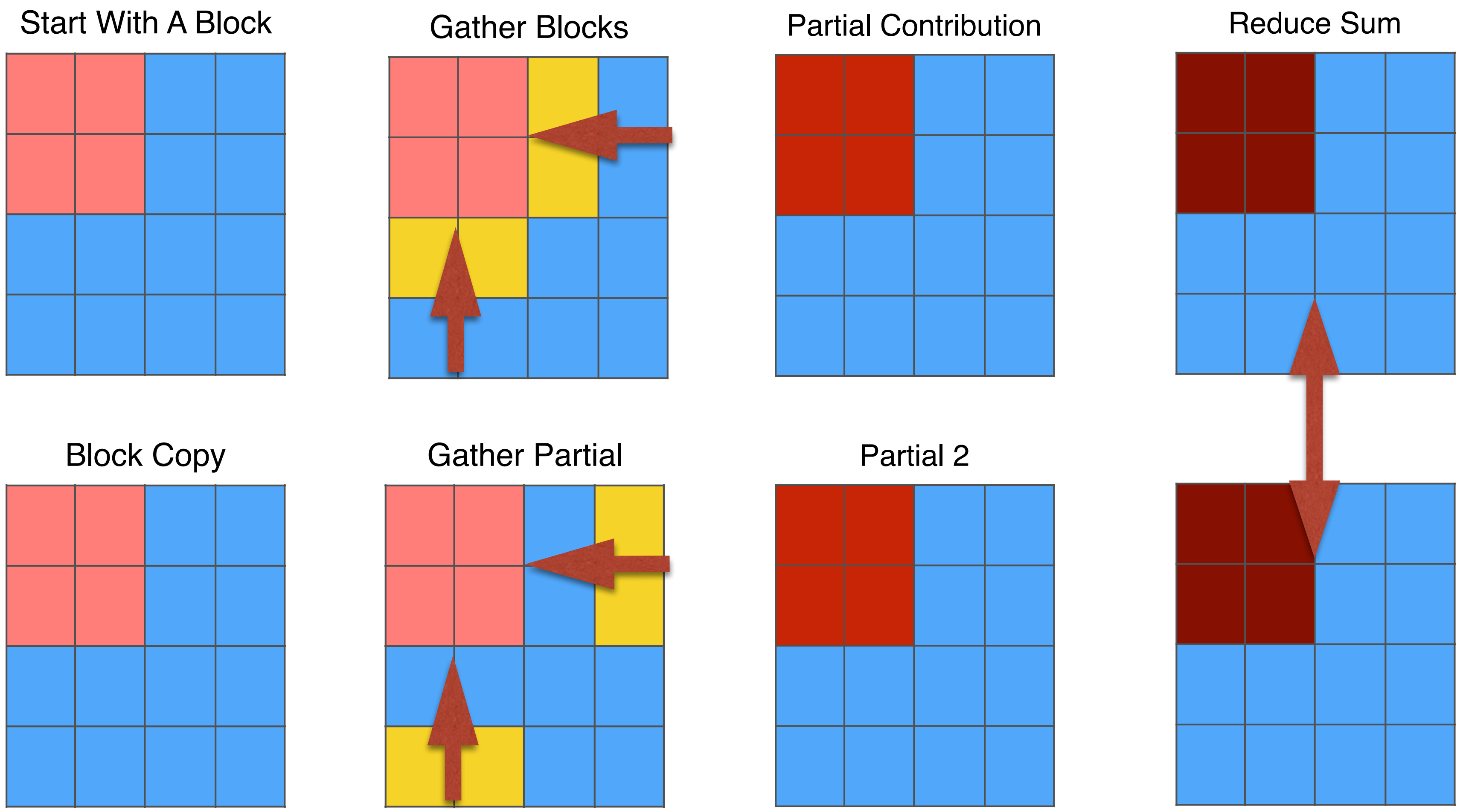
Compute Block



Schatz, Martin D., Robert A. Van de Geijn, and Jack Poulson. "Parallel matrix multiplication: A systematic journey." *SIAM Journal on Scientific Computing* 38, no. 6 (2016): C748-C781.

Matrix Multiplication Parallelization - 2.5D

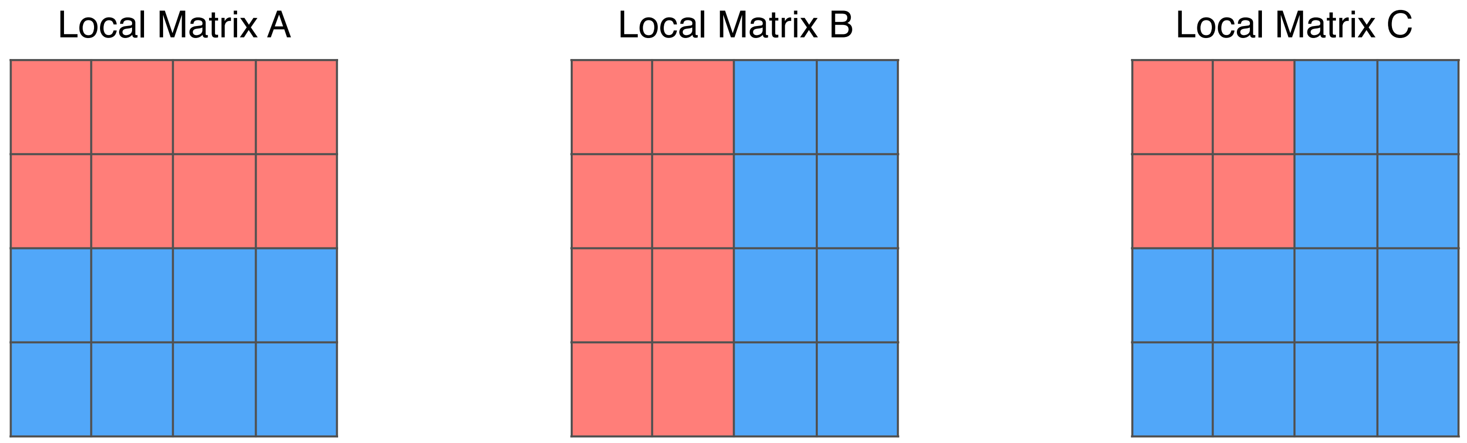
2.5D Algorithm: Duplicate In Z Direction



Solomonik, Edgar, and James Demmel. "Communication-optimal parallel 2.5 D matrix multiplication and LU factorization algorithms." In *European Conference on Parallel Processing*, pp. 90-109. Springer, Berlin, Heidelberg, 2011.

Matrix Multiplication Parallelization - OpenMP

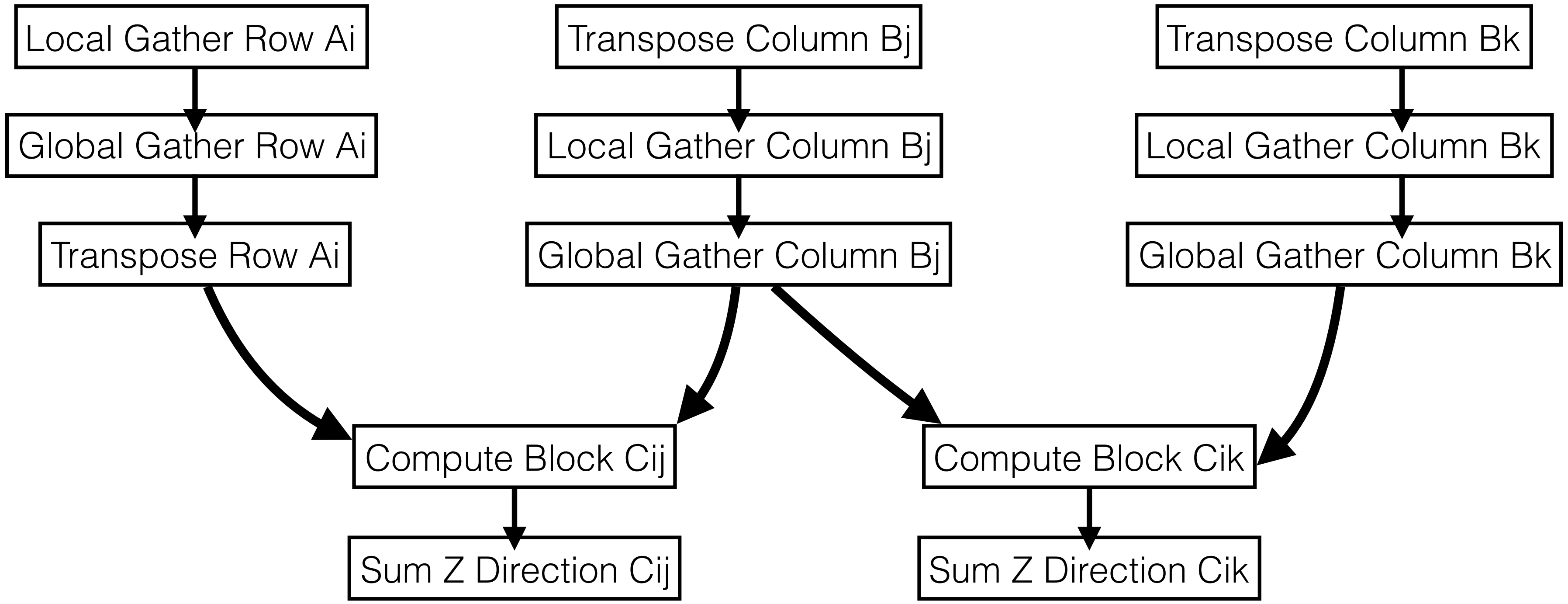
- Important to have a hybrid OpenMP/MPI implementation to target future architectures.
- Main idea: thread parallel over matrix blocks.
- Local blocked matrix multiply works like dense multiply.



- Can also block the communication, allowing for overlapping of communication and computation.
- Little overhead for blocking.

Matrix Multiplication Parallelization - OpenMP

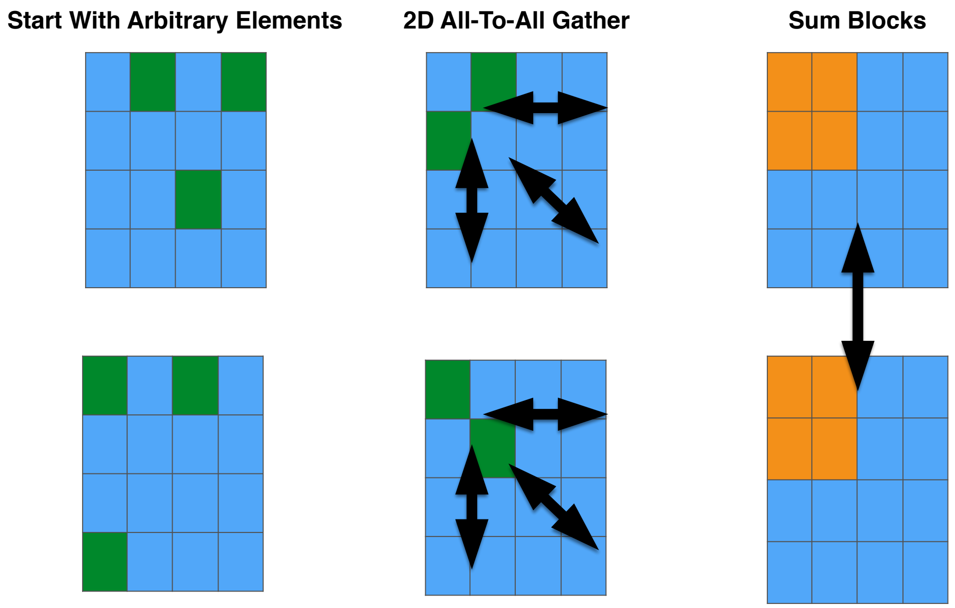
- OpenMP loop parallelism doesn't work well with overlapping communication. Instead we use OpenMP task framework.
- Creating a task manager, and dependency graph.

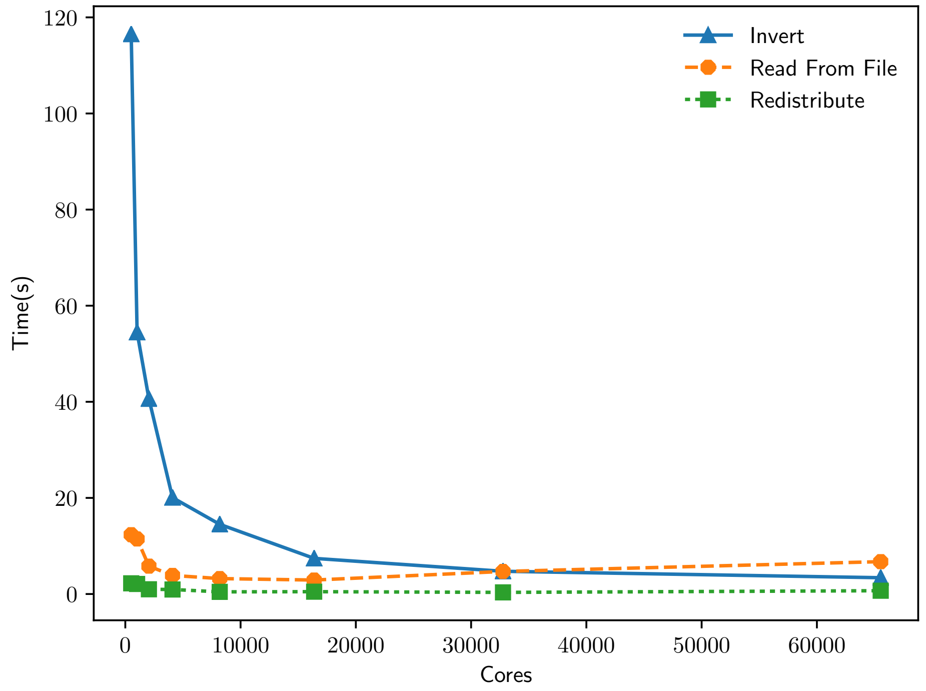


Usability Considerations

Usability Considerations

- Challenge: Integrate With Parallel Programs Using a Variety of Different Data Layouts.
- Solution 1: Parallel File I/O through the standard matrix market format for rapid prototyping.
- Solution 2: Arbitrary Data Remapping Routines.

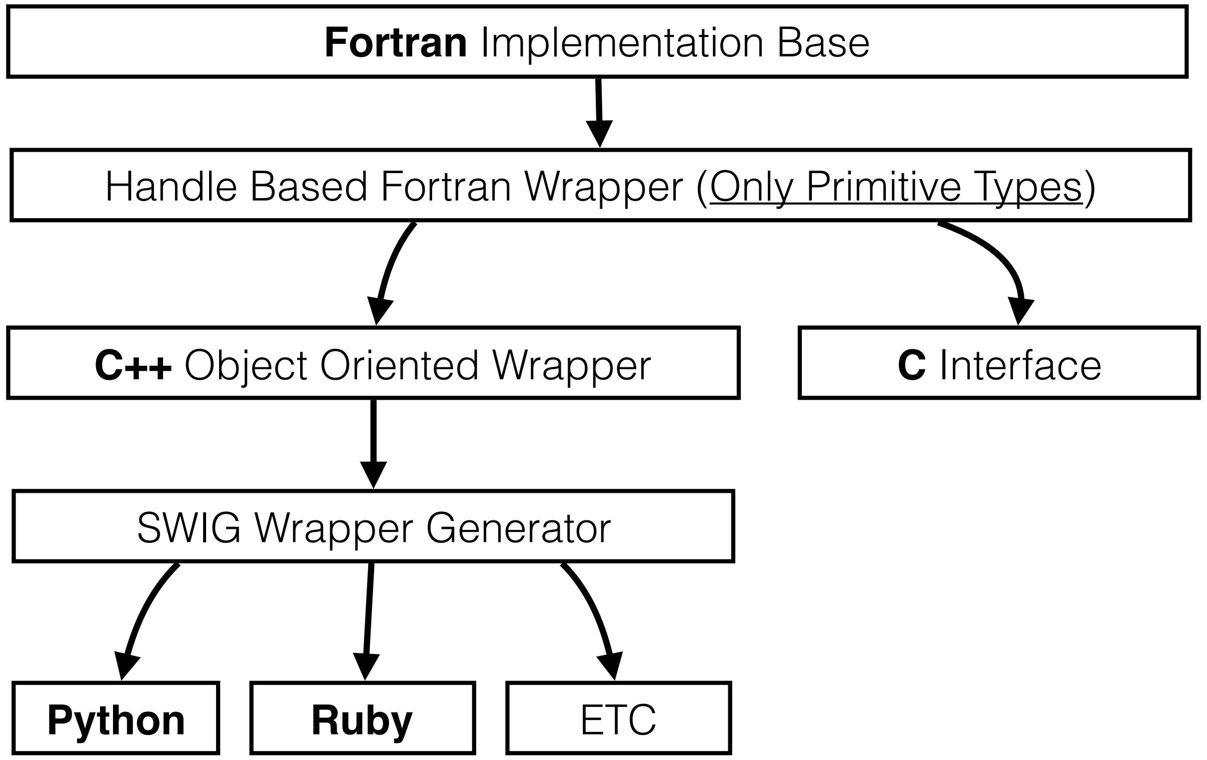




Usability Considerations - 2

Challenge: Integration with codes written in a variety of programming language.

Solution: Programming Language Wrapper Hierarchy.


Fortran Implementation Base
Handle Based Fortran Wrapper (Only Primitive Types)
C++ Object Oriented Wrapper
C Interface
SWIG Wrapper Generator
Python
Ruby
ETC

Programming Language Support - Details

Using complex data types makes life easier in Fortran, but makes it harder to call from other languages.

```
! Complex Fortran Data Type
TYPE :: DistributedSparseMatrix_t
   ! Simple data
   INTEGER :: matrix_dimension
   INTEGER :: start_column, end_column
   …
   ! Variety of members
   TYPE(ProcessGrid_t) :: grid
   ! Also contains allocatable subtypes
   TYPE(LocalMatrix_t), DIMENSION(:,:), ALLOCATABLE :: local_data
   …
END TYPE

SUBROUTINE ComputeExponential(InputMat, OutputMat)
    TYPE(DistributedSparseMatrix_t), INTENT(in)   :: InputMat
    TYPE(DistributedSparseMatrix_t), INTENT(inout) :: OutputMat
    ! Solver Logic
END SUBROUTINE
```

Pletzer, Alexander, Douglas McCune, Stefan Maszala, Srinath Vadlamani, and Scott Kruger. "Exposing Fortran derived types to C and other languages." *Computing in Science & Engineering* 10, no. 4 (2008): 86-92.

Programming Language Support - 2

To simplify things, we will only expose handles to data objects.

```
TYPE :: DistributedSparseMatrix_wrp ! Handle Datatype
   TYPE(DistributedSparseMatrix_t), POINTER :: DATA
END TYPE

SUBROUTINE ConstructMatrix_wrp(ih_this)
   INTEGER(kind=c_int), INTENT(INOUT) :: ih_this(SIZE_wrp) ! SIZE_wrp is size of a pointer struct.
   TYPE(DistributedSparseMatrix_wrp) :: this
   ALLOCATE(this%data)
   ih_this = TRANSFER(this,ih_this) ! Convert between handle and integer.
END SUBROUTINE

SUBROUTINE ComputeExponential_wrp(ih_InputMat, ih_OutputMat) bind(c,name=“ComputeExponential_wrp")
    INTEGER(kind=c_int), INTENT(in)    :: ih_InputMat(SIZE_wrp)
    INTEGER(kind=c_int), INTENT(inout) :: ih_OutputMat(SIZE_wrp)
    TYPE(DistributedSparseMatrix_wrp) :: InputMat
    TYPE(DistributedSparseMatrix_wrp) :: OutputMat

    InputMat = TRANSFER(ih_InputMat,InputMat)
    OutputMat = TRANSFER(ih_OutputMat,OutputMat)
    CALL ComputeExponential(InputMat%data, OutputMat%data)
END SUBROUTINE
```

Programming Language Support - 3

C Interface is now simple to expose.

```
// C Routine To Call
void ConstructMatrix_wrp(int *ih_this);
void ComputeExponential_wrp(const int *ih_Input, int *ih_Output);
```

C++ Uses the same interface

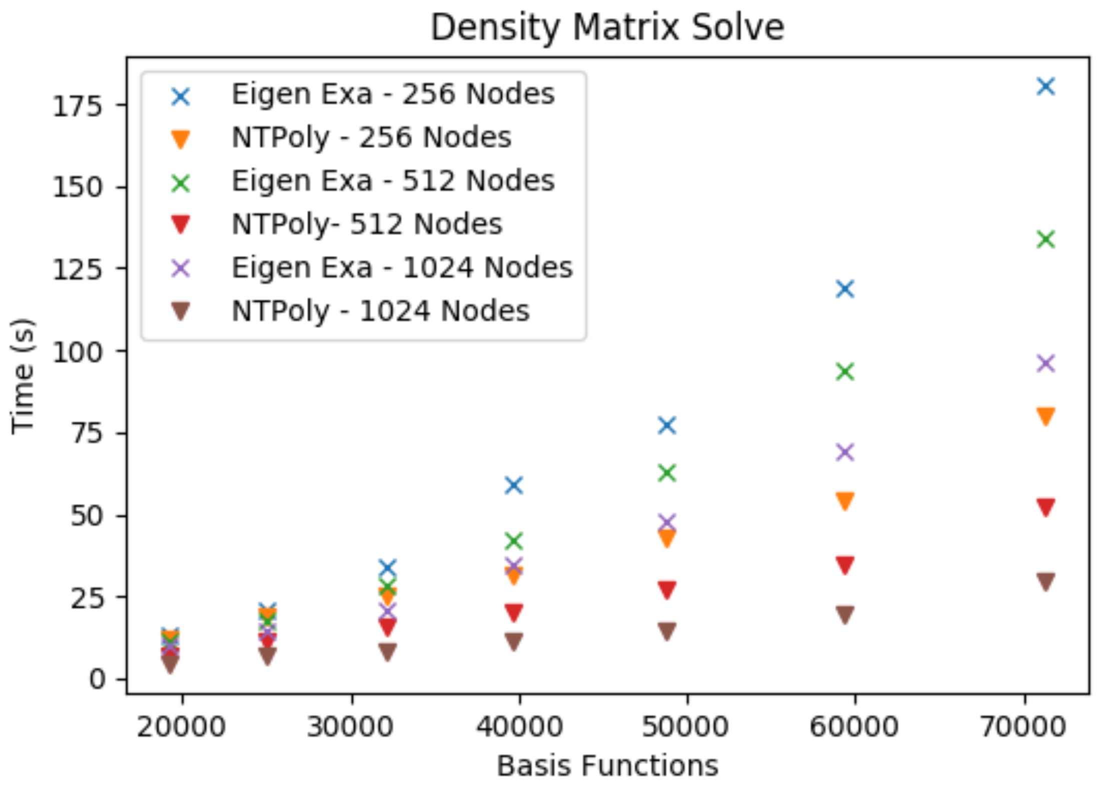
```
class DistributedSparseMatrix {
public:
  DistributedSparseMatrix() {
    ConstructMatrix_wrp(this->handle);
  }
  int handle[SIZE_wrp];
};

void ComputeExponential(const DistributedSparseMatrix &InputMat,
       DistributedSparseMatrix &OutputMat) {
   ComputeExponential_wrp(InputMat.handle, OutputMat.handle);
}
```

Example Applications

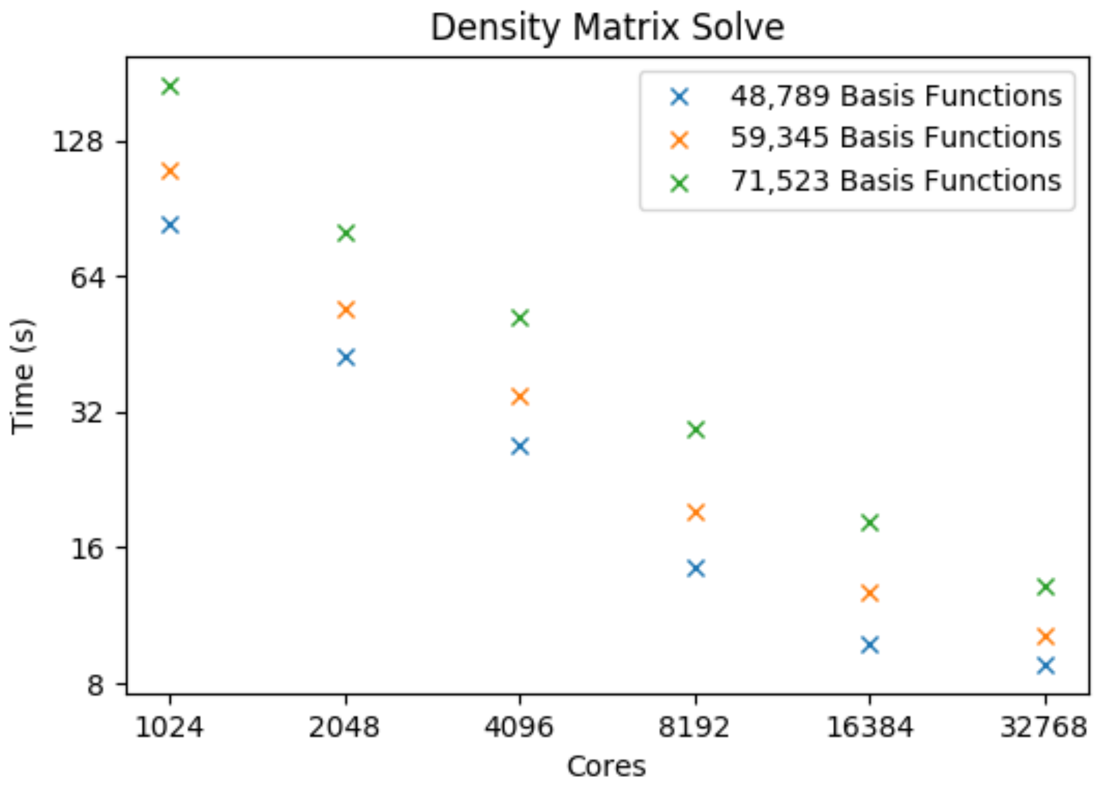
Quantum Chemistry

- Standard eigensolvers can make limited use of the sparsity of a matrix, but will be outperform by matrix function based approaches.
- Calculation of water clusters of various sizes, 631G basis set, using the TRS2 density matrix method to approximate the fermi function.



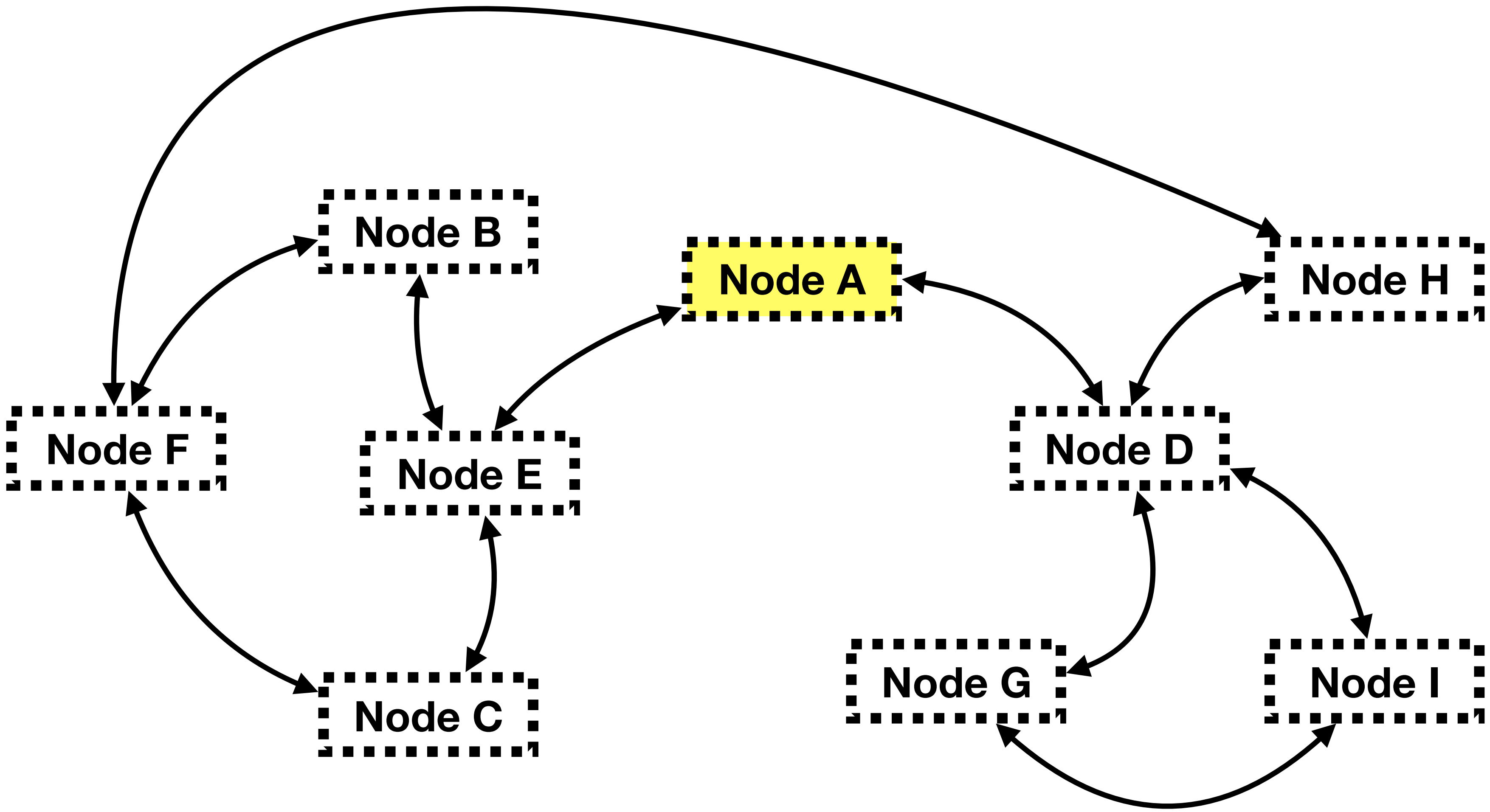
Quantum Chemistry - 2

- Use of communication avoiding algorithms and task based openmp parallelization allows for calculations using tens of thousands of cores.



Social Network Analysis

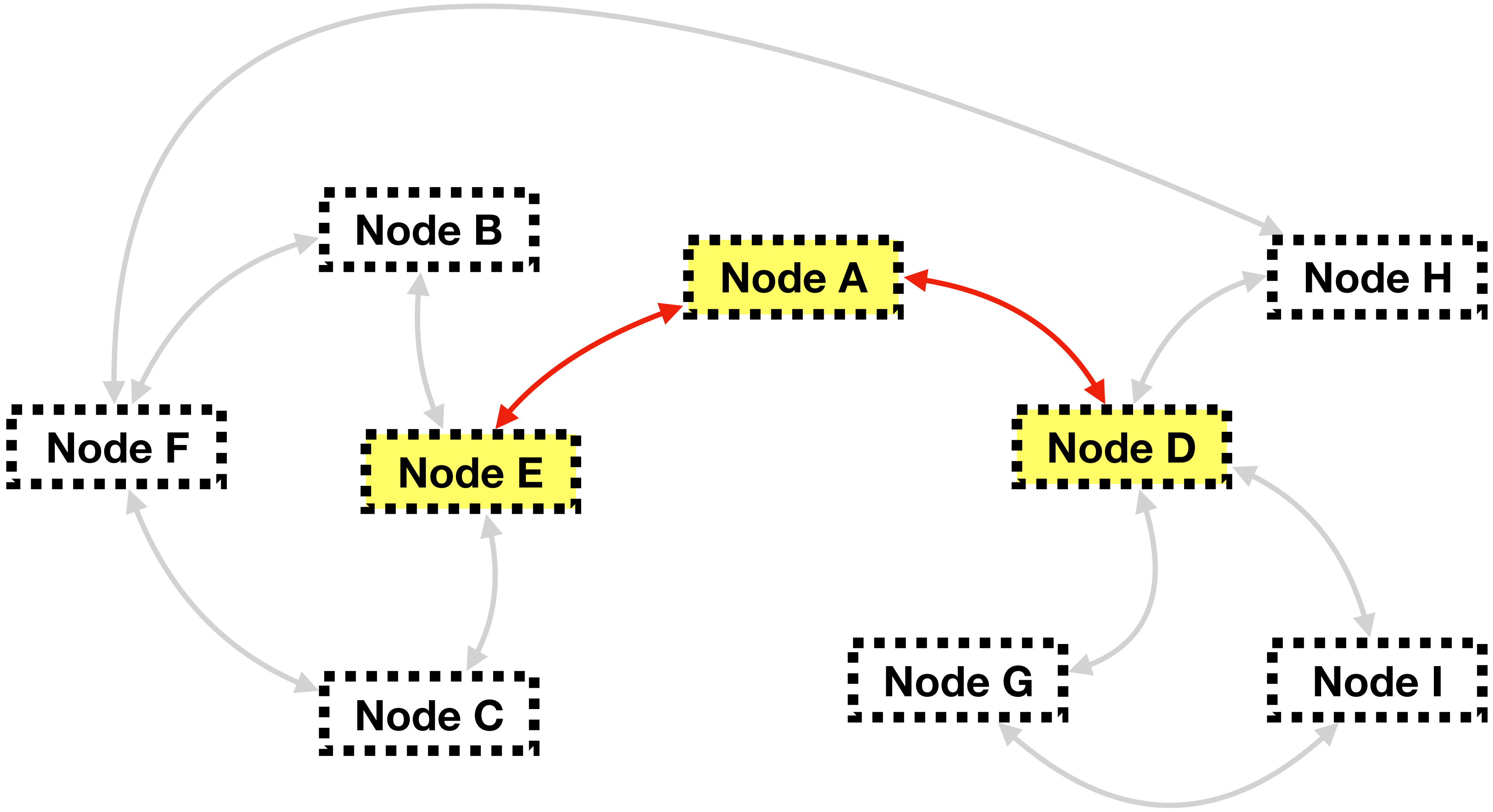
- Estrada's Scaled Matrix Exponential Metric: $e^{\beta A}$
- Example, social networks:



Estrada, Ernesto, Naomichi Hatano, and Michele Benzi. "The physics of communicability in complex networks." Physics reports 514, no. 3 (2012): 89-119. 29

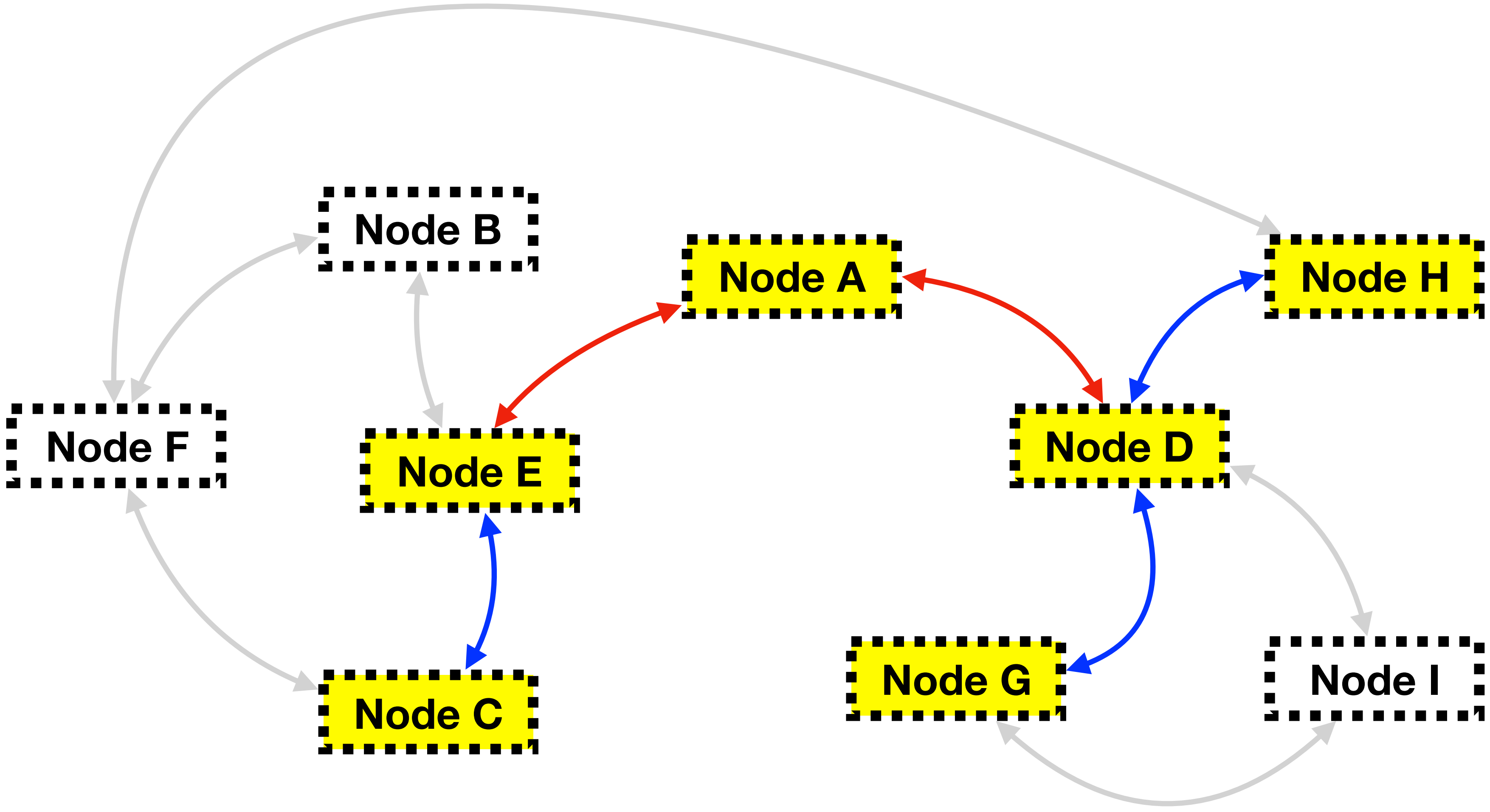
Social Network Analysis

- Estrada's Scaled Matrix Exponential Metric: $e^{\beta A}$
- Example, social networks:



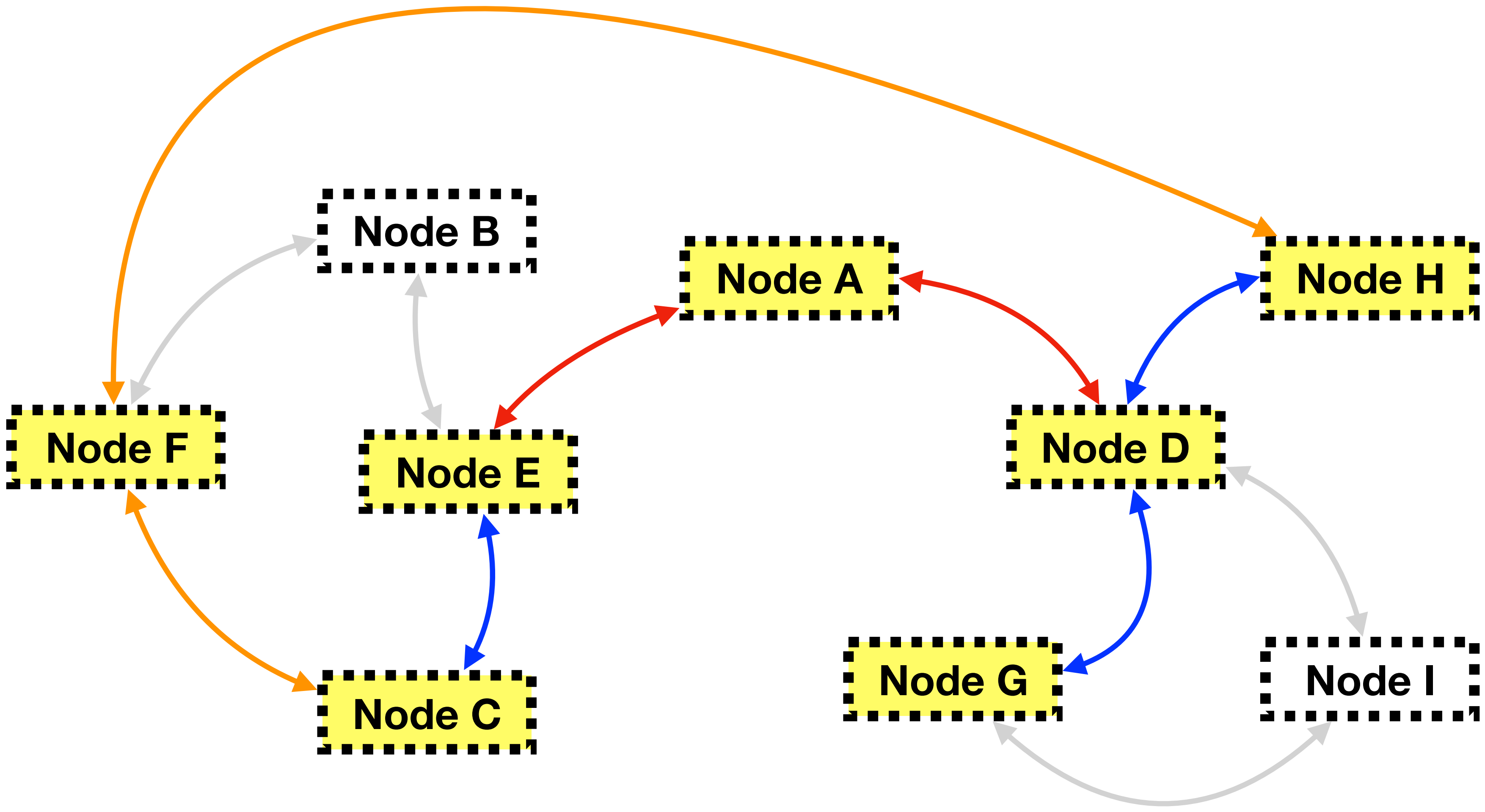
Social Network Analysis

- Estrada's Scaled Matrix Exponential Metric: $e^{\beta A}$
- Example, social networks:



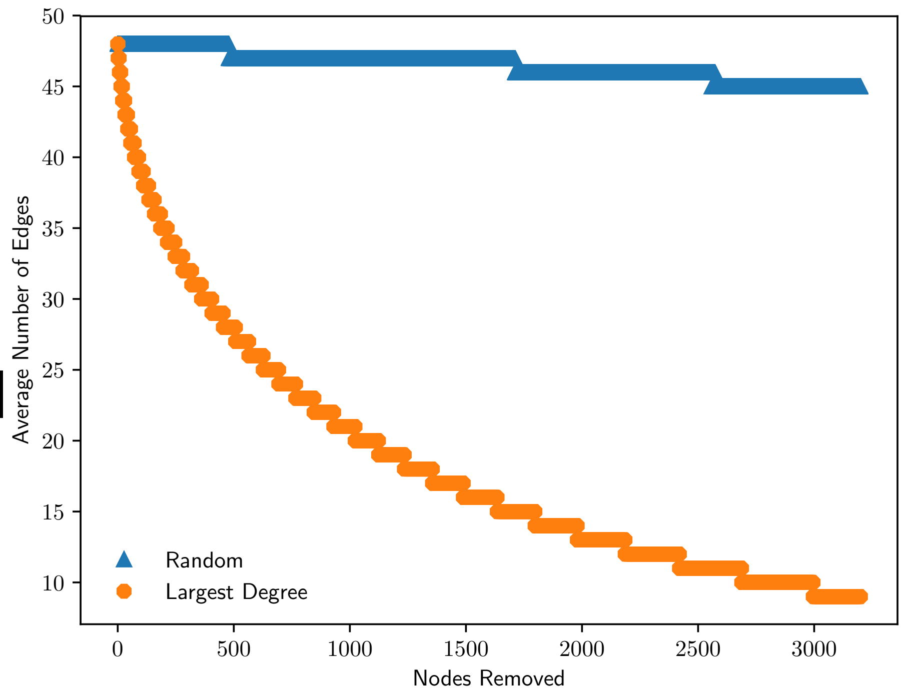
Social Network Analysis

- Estrada's Scaled Matrix Exponential Metric: $e^{\beta A}$
- Example, social networks:



Social Network Analysis - In Practice

- Network Resiliency calculations:
  - Data Set: Israeli Social Network “TheMarker Cafe”
  - Nodes: 69413. Sparsity: 0.04%.
- Procedure:
  - Remove a node from the graph
    - Random Node
    - Node with largest degree
  - Compute the matrix exponential
  - Compute the sparsity
  - Repeat



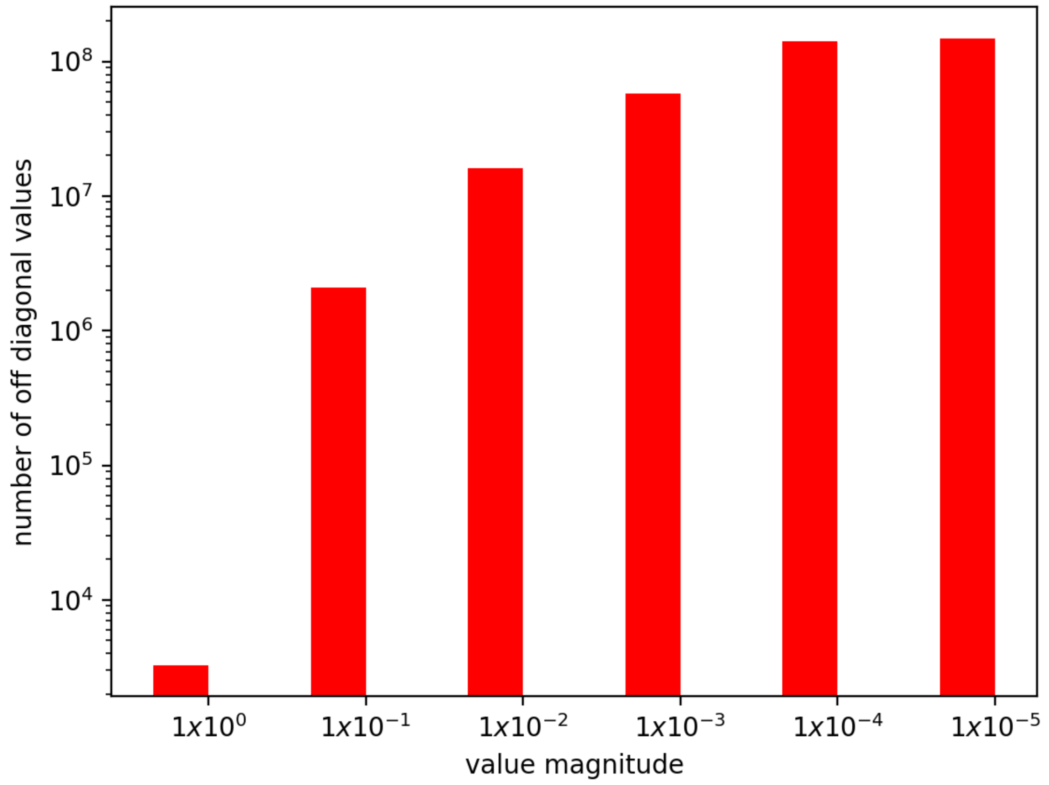
Search Engine Optimization

- Finding important nodes in a directed network.
- Going beyond page rank by ranking nodes as Authorities and Hubs. Authorities are important nodes, hubs point to authorities.
- Matrix functions to compute:
  - Hub similarity matrix: $H = \cosh(\sqrt{(AA^T)})$.
  - Authority similarity matrix: $A = \cosh(\sqrt{(A^TA)})$.
- Calculation Method:
  - Scaling and squaring method with Chebyshev polynomials: $\cosh(\sqrt{4x}) = 2\cosh^2\sqrt{x} - 1$.

Benzi, Michele, Ernesto Estrada, and Christine Klymko. "Ranking hubs and authorities using matrix functions." Linear Algebra and its Applications 438, no. 5 (2013): 2447-2474.

<u>Search Engine Optimization - 2</u>

- Authority similarity for the High Energy Physics Phenomenology citation graph.
- 34,546 papers, 421,578 citations.
- Largest eigenvalue is > 3000, so some scaling factor is necessary.
- Sparsity:
  - A: ~0.04%.
  - AA^T: ~1%
  - $\cosh(\sqrt{(\beta AA^T)})$: 31%.



Conclusion

- Matrix functions have a number of different applications, motivating the creation of new libraries.
- Using sparse matrix algebra techniques, low order scaling techniques exist to compute the functions of sparse matrices.
- A combination of communication avoiding algorithms and task based parallelization enable strong scaling, massively parallel calculations.
- Automated data redistribution techniques, parallel file i/o using standard formats, and support for many programming languages leads to an easy to use library.
- Applications to quantum chemistry, graph analysis, and more.
- https://william-dawson.github.io/NTPoly/