The development of the electronic structure code LSDalton and its Library Requirements

Thomas Kjærgaard

LSDalton

Electronic Structure Program

- Mean-Field Self-Consistent Field (SCF)
- Electron Correlation: Divide-Expand-Consolidate (DEC)

- Source code free of charge daltonprogram.org
- Research tool

Overview

- The LSDalton program
- The SCF module
  - Design
- The DEC module
  - Design
  - Parallelization
  - Performance

LSDalton

- Fortran90
- MPI/OpenMP hybrid
- Platform independent
- Support multiple compilers:
  - Intel, GNU, PGI, Cray, XL

LSDalton

- Developed by domain scientists
  - Focus on research NOT implementation
  - Written by Ph.D. stud. & Post Docs. Fast turn around
  - Code easy to read, extend, and maintain
  - 80 % efficiency is good enough.
  - Speed through LAPACK & other Libraries
  - Minimum of programing languages: Fortran, MPI, OpenMP, OpenACC
  - No CUDA, Fortran2008, Coarray Fortran, C, C++

LSDalton: OpenMP/OpenACC

- Domain scientists
  - Easy to write and extend
  - Easy to maintain (by others)
  - Single source code
  - No need to reoptimize (performance portability)
  - Speed through CuBLAS/libsci_acc & other Libraries
  - Works with all accelerators (NVIDIA GPUs, non NVIDIA GPUs, Xeon Phi)
  - Limited to PGI and Cray! Problem with testing!
  - Multiple GPUs “PGI vs Cray”

GPU lessons learned

- CUDA vs OpenACC vs OpenMP
- GPU enabled libraries easy and fast
- Data traffic important
- GPU memory management
- Tools of the trade cuda-memcheck, cuda-gdb, etc.
- Easy to include OpenACC at design phase

SCF

$$F(C) \cdot C = S \cdot C \cdot \epsilon$$

Generalized eigenvalue equation

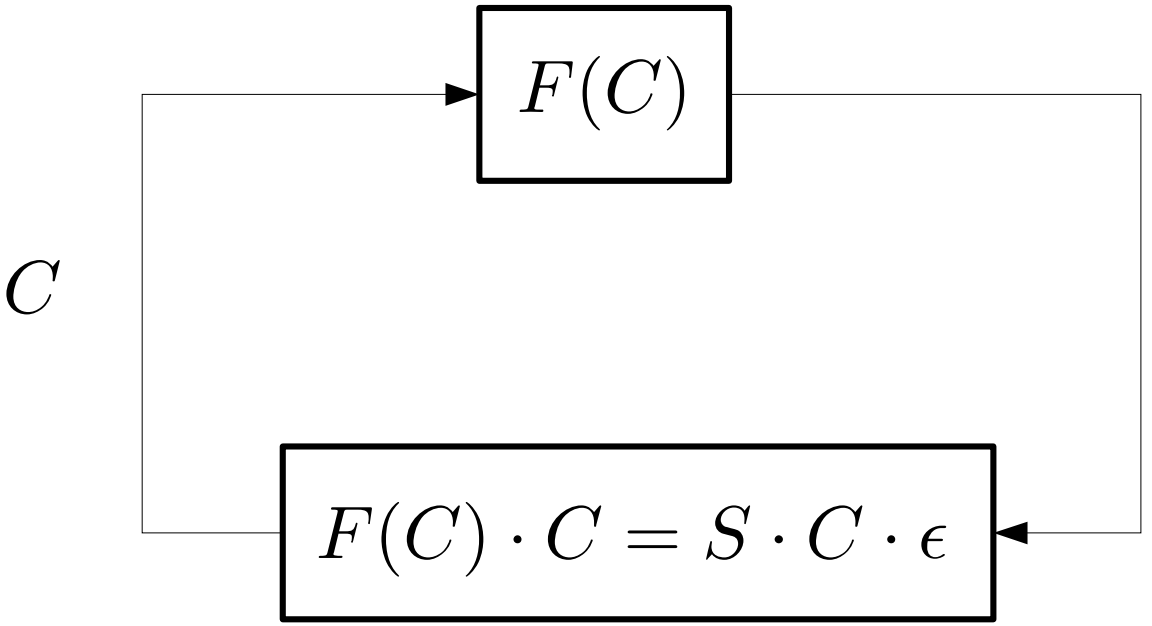
$F(C)$ Depend on the Solution!

SCF

$$F(C) \cdot C = S \cdot C \cdot \epsilon$$

Generalized eigenvalue equation

$F(C)$ Depend on the Solution!

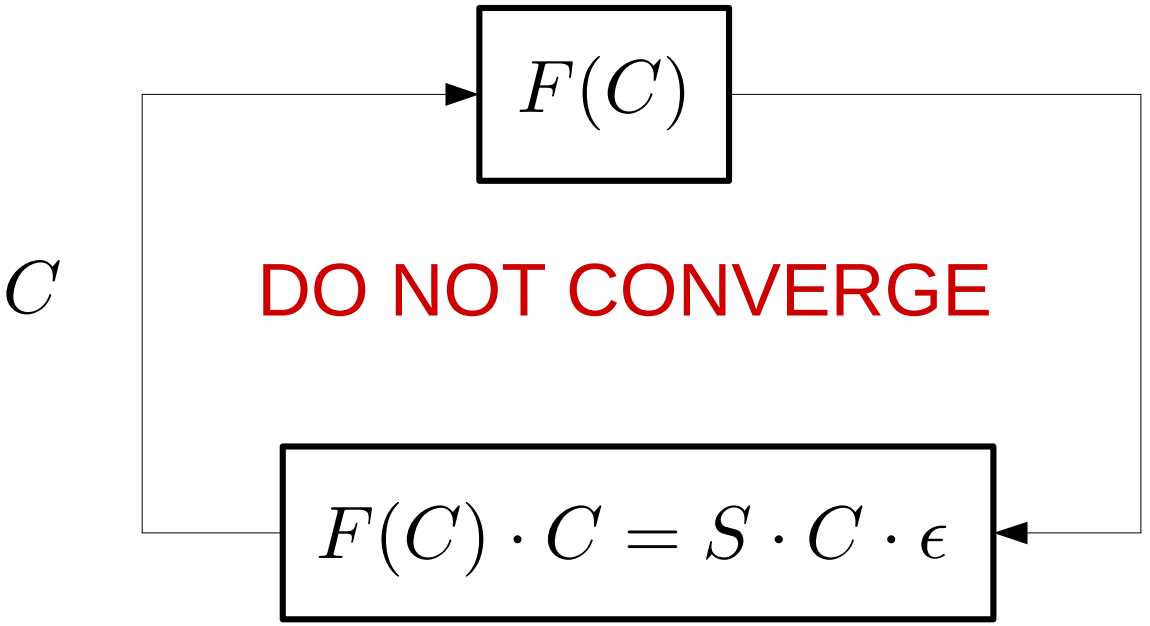


SCF

$$F(C) \cdot C = S \cdot C \cdot \epsilon$$

Generalized eigenvalue equation

$F(C)$ Depend on the Solution!



SCF

$$F(C) \cdot C = S \cdot C \cdot \epsilon$$

Replaced by a number of matrix multiplications

$$F(C)$$

Complicated and requires the evaluation of:

$$\int \int \frac{\phi_a(r_1)\phi_b(r_1)\phi_c(r_2)\phi_d(r_2)}{|r_1 - r_2|} dr_1 dr_2$$

Domain science specific

SCF

$$F(C) \cdot C = S \cdot C \cdot \epsilon$$

Replaced by a number of matrix multiplications

- 2 dimension quantities (matrices)
- Mainly matrix matrix multiplications

SCF

$$F(C) \cdot C = S \cdot C \cdot \epsilon$$

Replaced by a number of matrix multiplications

Current:

Dimensions: 30.000 x 30.000 (~7 GB per matrix)
30 matrices ⟶ 216 GB

Goal:

Dimensions: 80.000 x 80.000 (~51 GB per matrix)
30 matrices ⟶ 1536 GB

Parallel Distributed Memory (not a single matrix in full)
Not ported to GPU
Calculations done on local cluster

SCF

SCF Wish List:

- Parallel Distributed matrices (PDM) ScaLAPACK
- MPI parallel calculation (with threaded BLAS) ScaLAPACK/Cray or ScaLAPACK/MKL
- GPU acceleration (multiple GPUs) MPI/CuBLAS
  - Hide data transport with sequence of matrix operations
- Sparsity (Highly system dependent 5%~40%) Automatic exploitation

  or Some PDM Block Sparse format
- Limited LAPACK, Full BLAS support, Hadamard product, extract diagonal, extract block, add block, etc.
- Fortran90 interface (*easy*, invest time with no obvious scientific paper)

Electron Correlation

$$E = \sum_{ij} \sum_{ab} t_{ij}^{ab} (2 g_{aibj} - g_{biaj})$$

$$0 = g_{aibj} + \sum_{c} \left(t_{ij}^{cb} F_{ac} + t_{ij}^{ac} F_{bc} \right) - \sum_{k} \left(t_{kj}^{ab} F_{ki} + t_{ik}^{ab} F_{kj} \right)$$

- 4 dimension quantities
- Contraction of 4 dimension quantities
- Set of non linear equations
- Huge memory requirements
- Requires an preliminary SCF calculation

Electron Correlation

$$E = \sum_{ij} \sum_{ab} t_{ij}^{ab} (2 g_{aibj} - g_{biaj})$$

$$0 = g_{aibj} + \sum_{c} \left(t_{ij}^{cb} F_{ac} + t_{ij}^{ac} F_{bc} \right) - \sum_{k} \left(t_{kj}^{ab} F_{ki} + t_{ik}^{ab} F_{kj} \right)$$

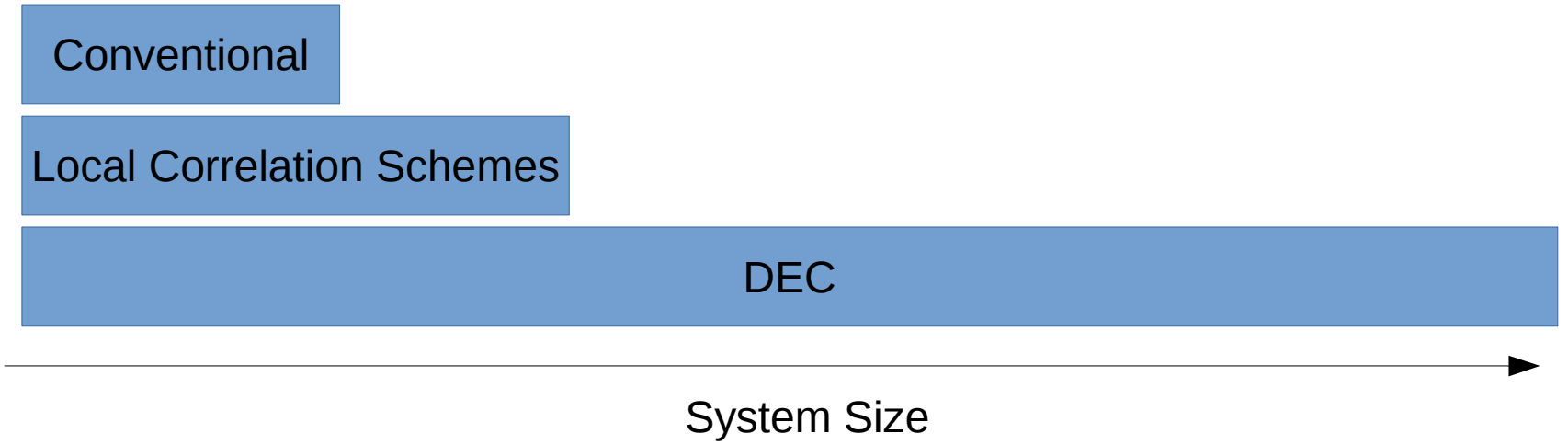
Largest calculation we have done:
Number of Occupied = 4760
Number of Virtual = 19680

70202 TB

PDM not enough! Disk Space not enough!
Conventional approach is not enough!

Divide-Expand-Consolidate

- Develop with Oak Ridge National Laboratory
  - INCITE 2015
  - INCITE 2016
  - CAAR
  - User meeting



Divide-Expand-Consolidate

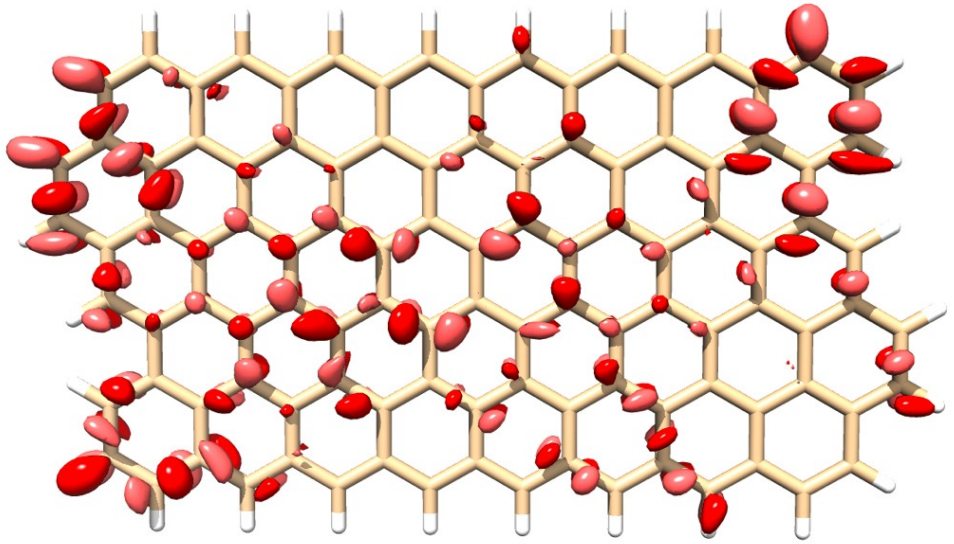
Correlated calculations describe a correction to SCF

LOCAL electron correlation effects:

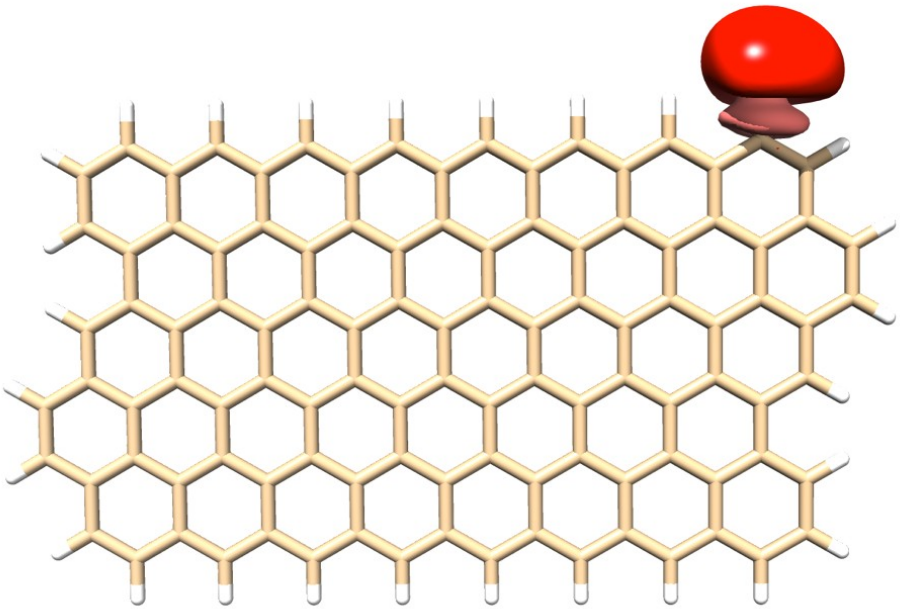
Short-range electron repulsion

Dispersion effects

Canonical (conventional) basis

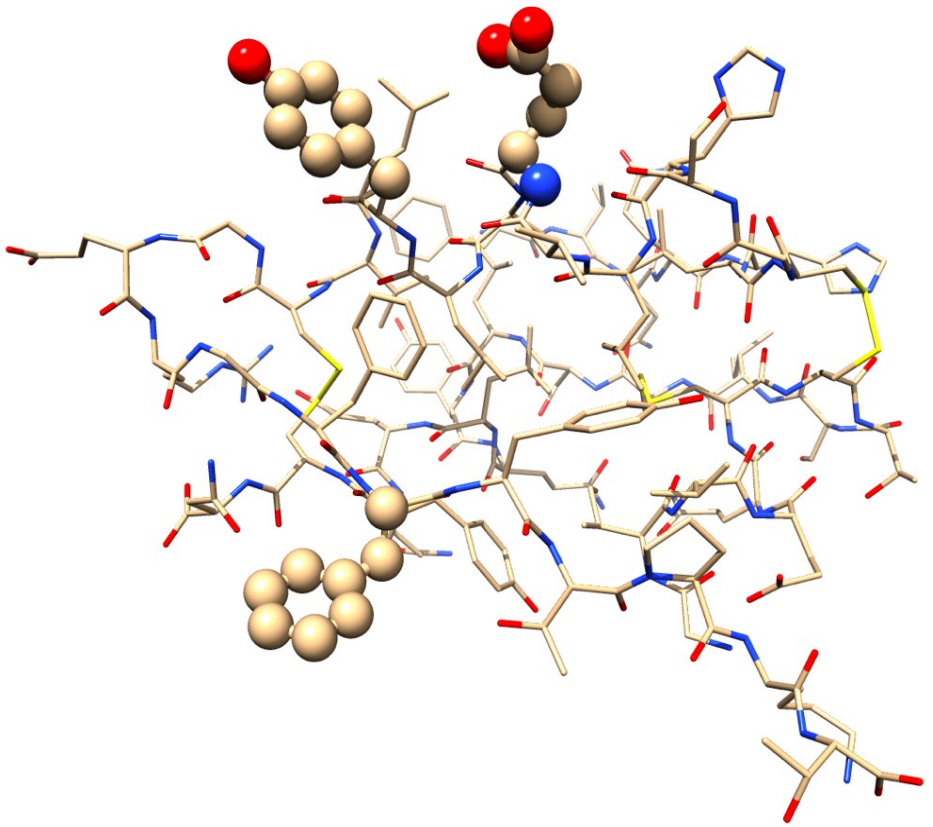


Local basis



We use a Local basis to describe the local correlation effect

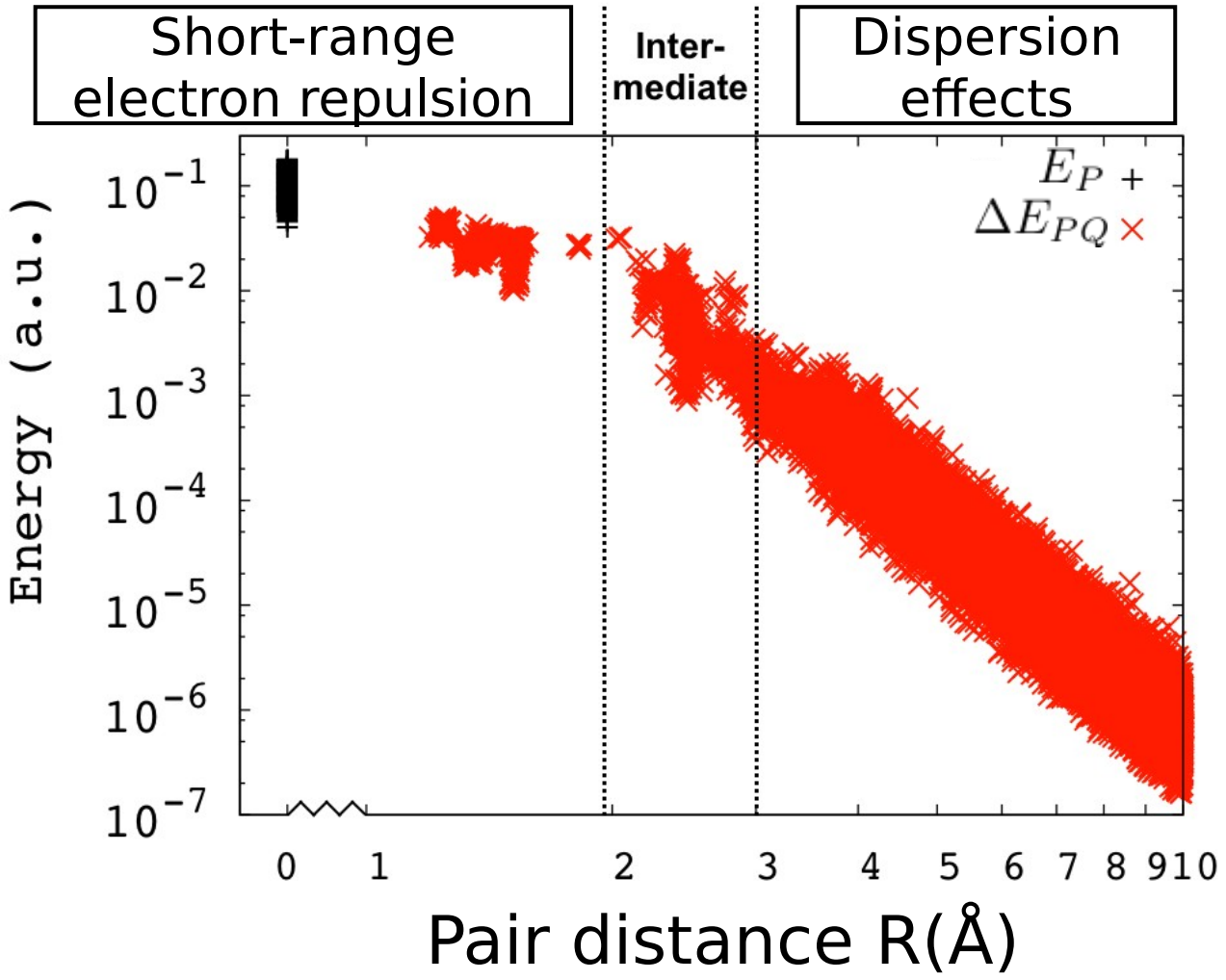
Divide-Expand-Consolidate



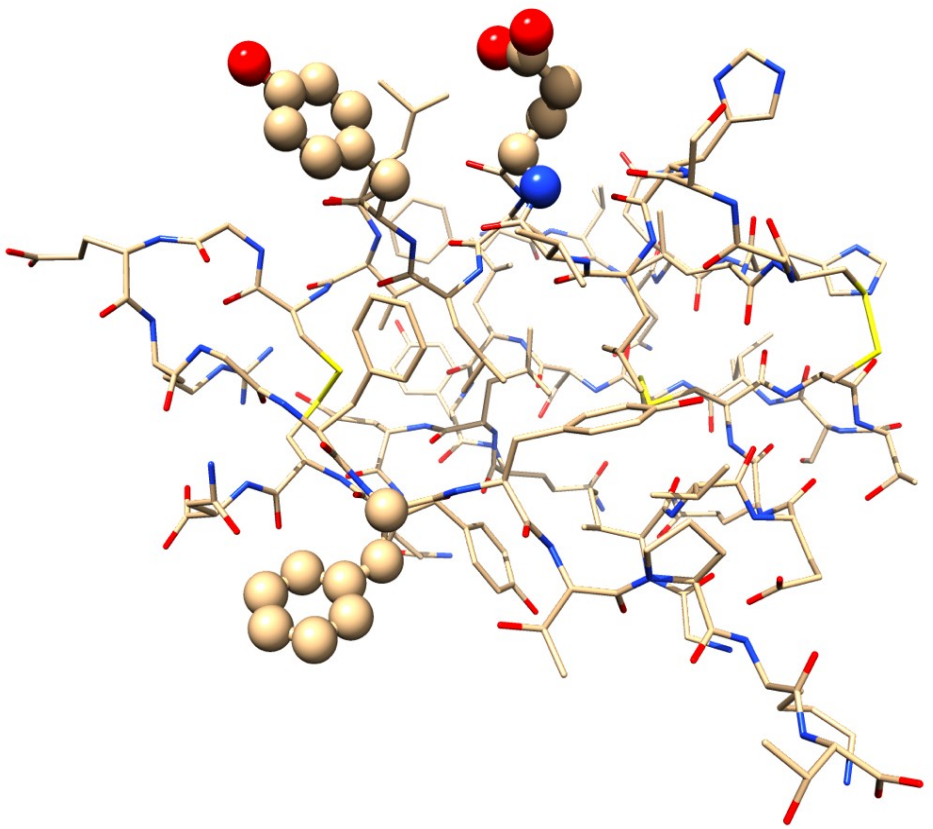
- Energy partitioning
- Atomic fragment calculations
- Pair fragment calculations
- Buffer space

Recalculation

Divide-Expand-Consolidate



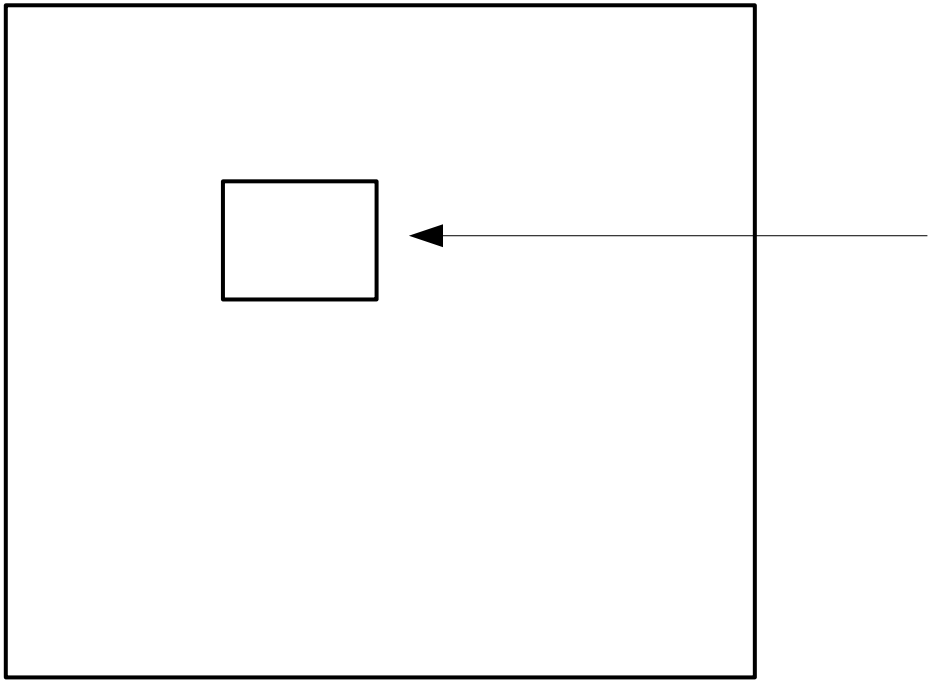
Divide-Expand-Consolidate



- Physical interpretation of pairs
- Screen away pairs
- Independent

Divide-Expand-Consolidate

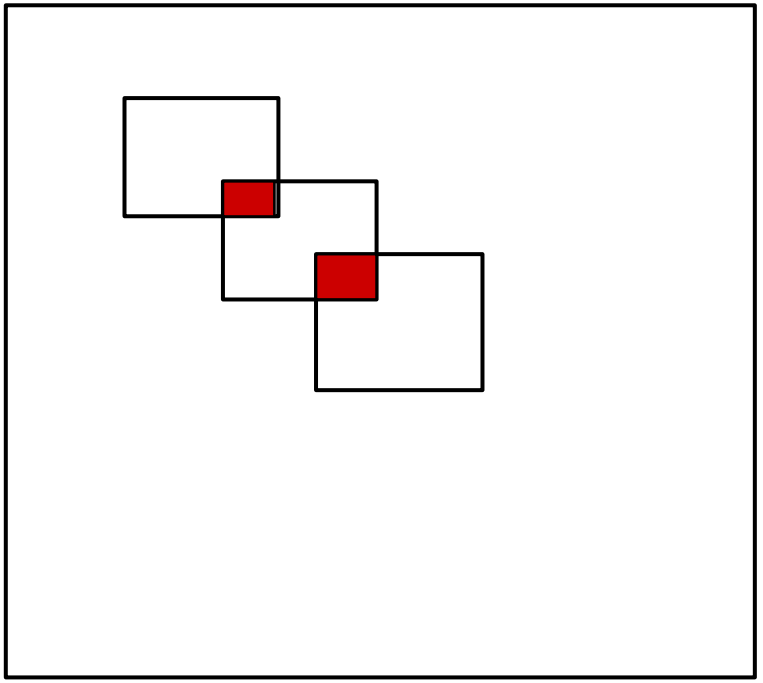
$t_{IJ}^{AB} =$



$$0 = g_{aibj} + \sum_{c} \left(t_{ij}^{cb} F_{ac} + t_{ij}^{ac} F_{bc} \right) - \sum_{k} \left(t_{kj}^{ab} F_{ki} + t_{ik}^{ab} F_{kj} \right)$$

Divide-Expand-Consolidate

$t_{IJ}^{AB} =$

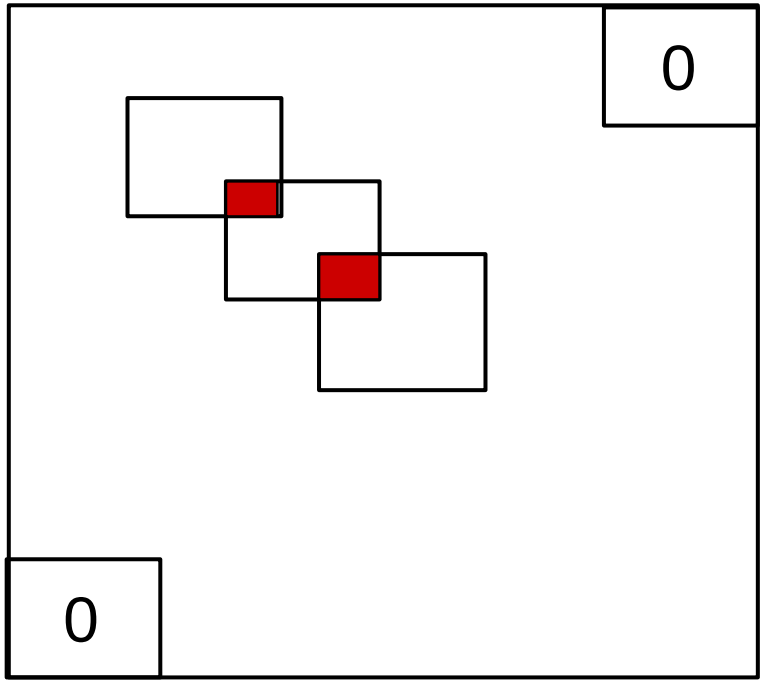


Overlapping
Fragment Calculation
Recalculation

$$0 = g_{aibj} + \sum_{c} \left(t_{ij}^{cb} F_{ac} + t_{ij}^{ac} F_{bc} \right) - \sum_{k} \left(t_{kj}^{ab} F_{ki} + t_{ik}^{ab} F_{kj} \right)$$

Divide-Expand-Consolidate

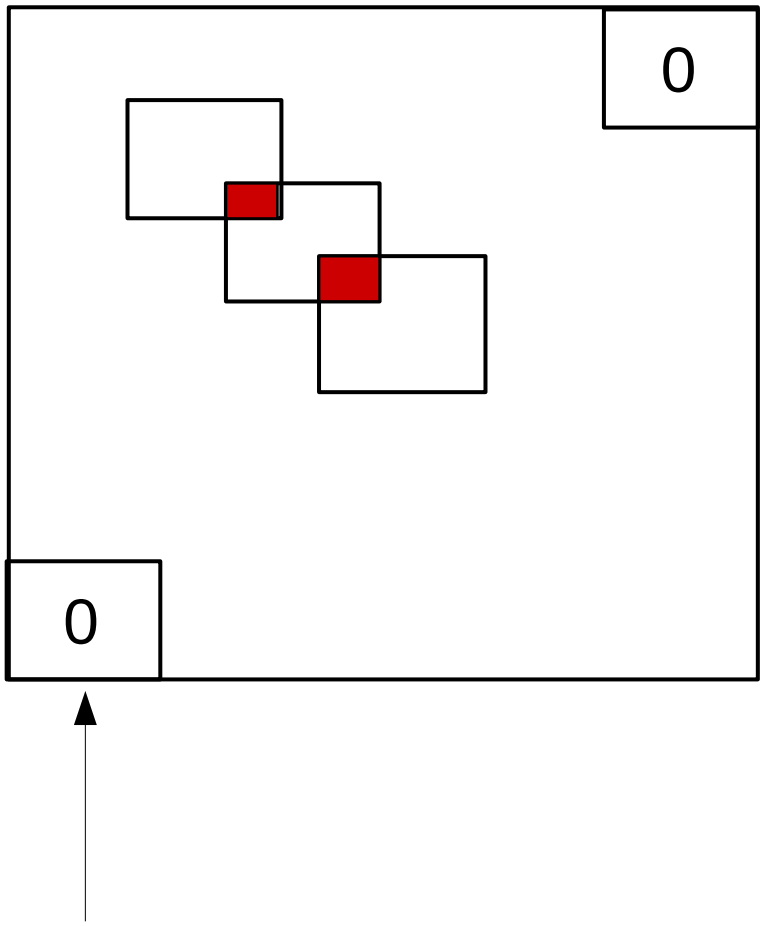
$t_{IJ}^{AB} =$



$$0 = g_{aibj} + \sum_{c} \left(t_{ij}^{cb} F_{ac} + t_{ij}^{ac} F_{bc} \right) - \sum_{k} \left(t_{kj}^{ab} F_{ki} + t_{ik}^{ab} F_{kj} \right)$$

Divide-Expand-Consolidate

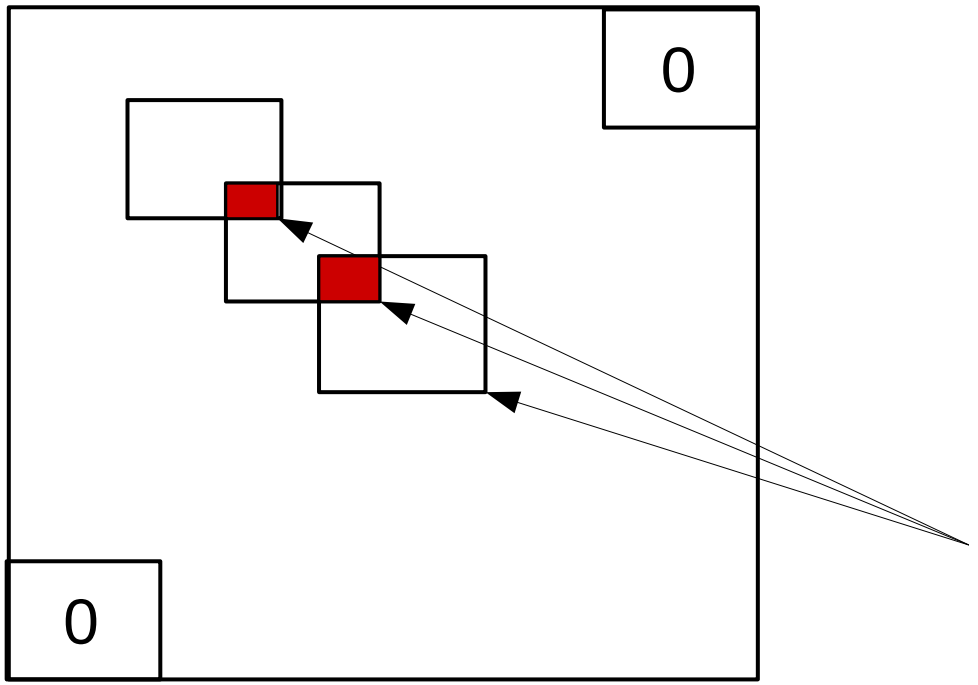
$$t_{IJ}^{AB} =$$



Exploit sparsity
Linear scaling

Divide-Expand-Consolidate

$$t_{IJ}^{AB} =$$



Independent Calculations
Massively parallel

Exploit sparsity
Linear scaling

DEC parallelization

Coarse grained level:

- MPI dynamic parallel distribution of fragments to local masters

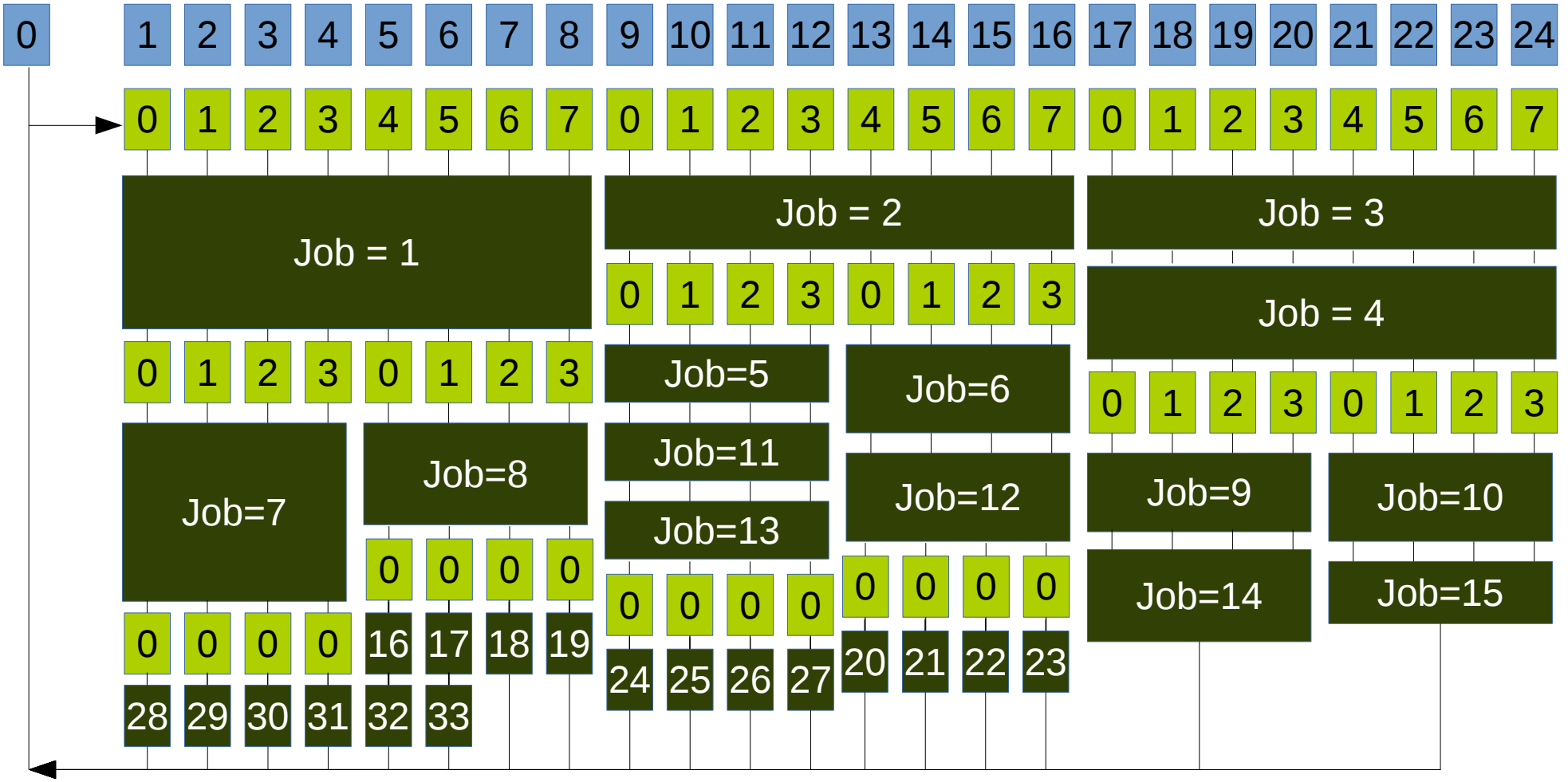
Medium grained level:

- MPI parallel fragment calculation

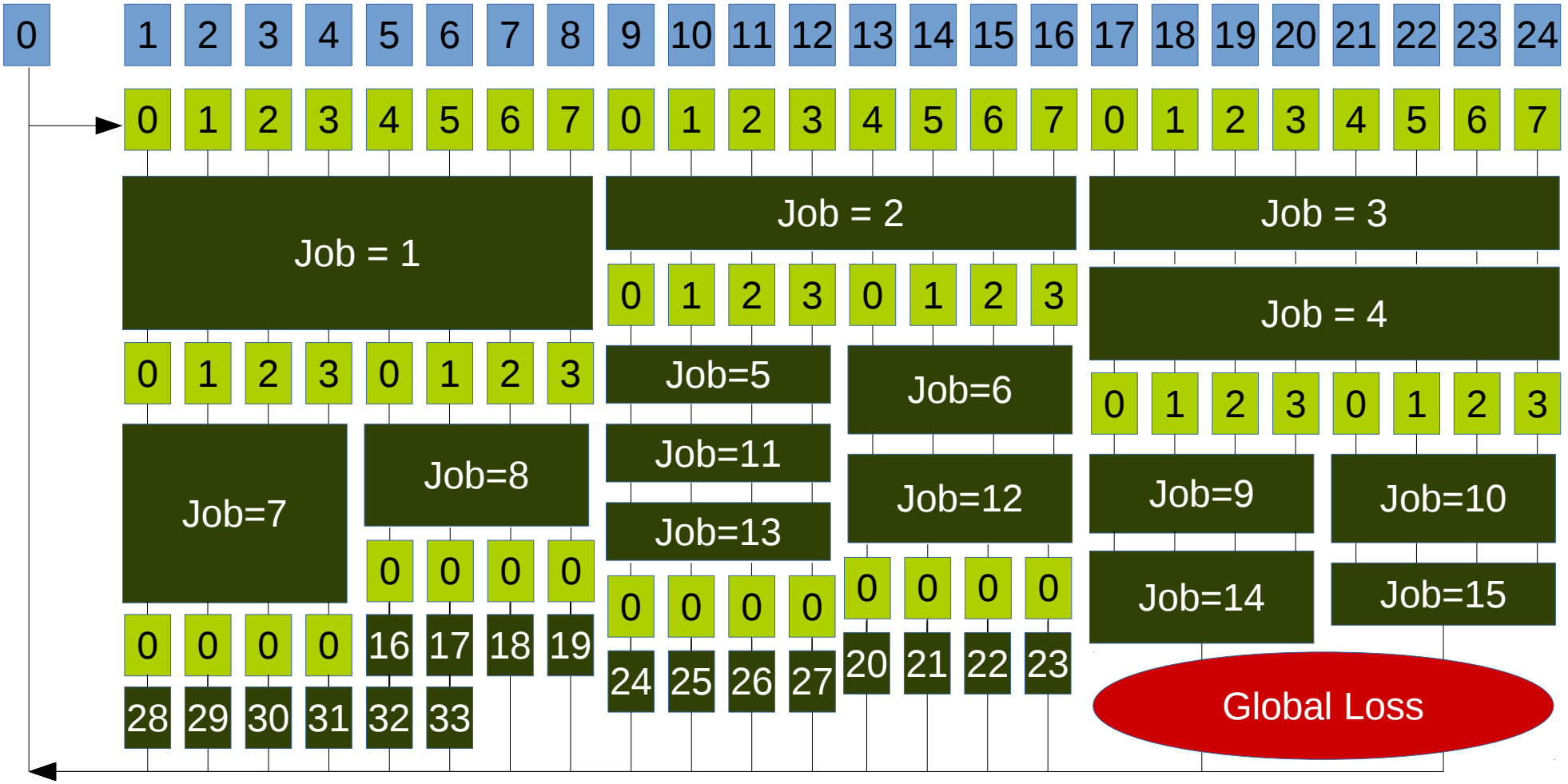
Fine grained level

- OpenMP/threaded BLAS
- OpenACC/CuBLAS

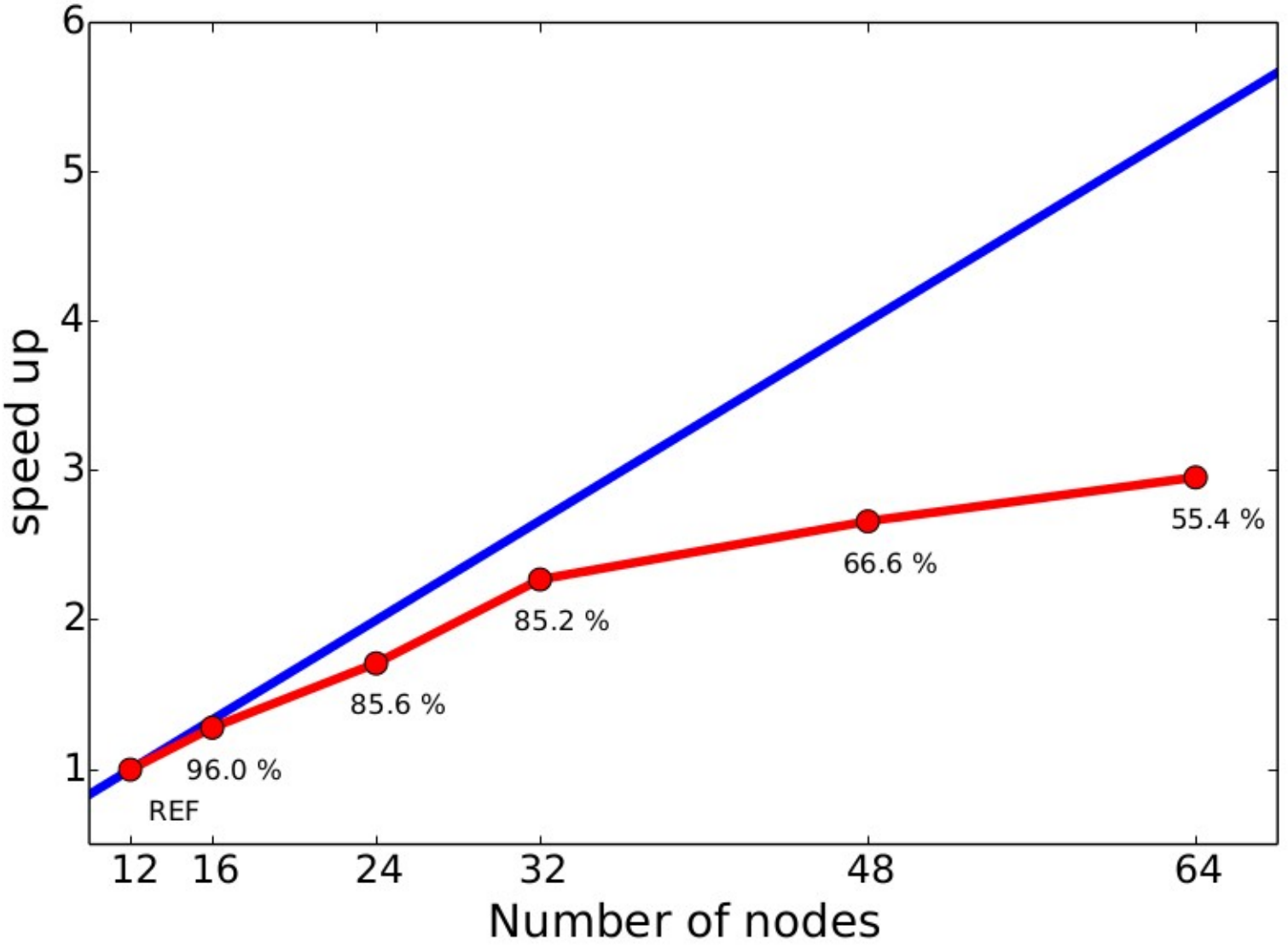
DEC parallelization



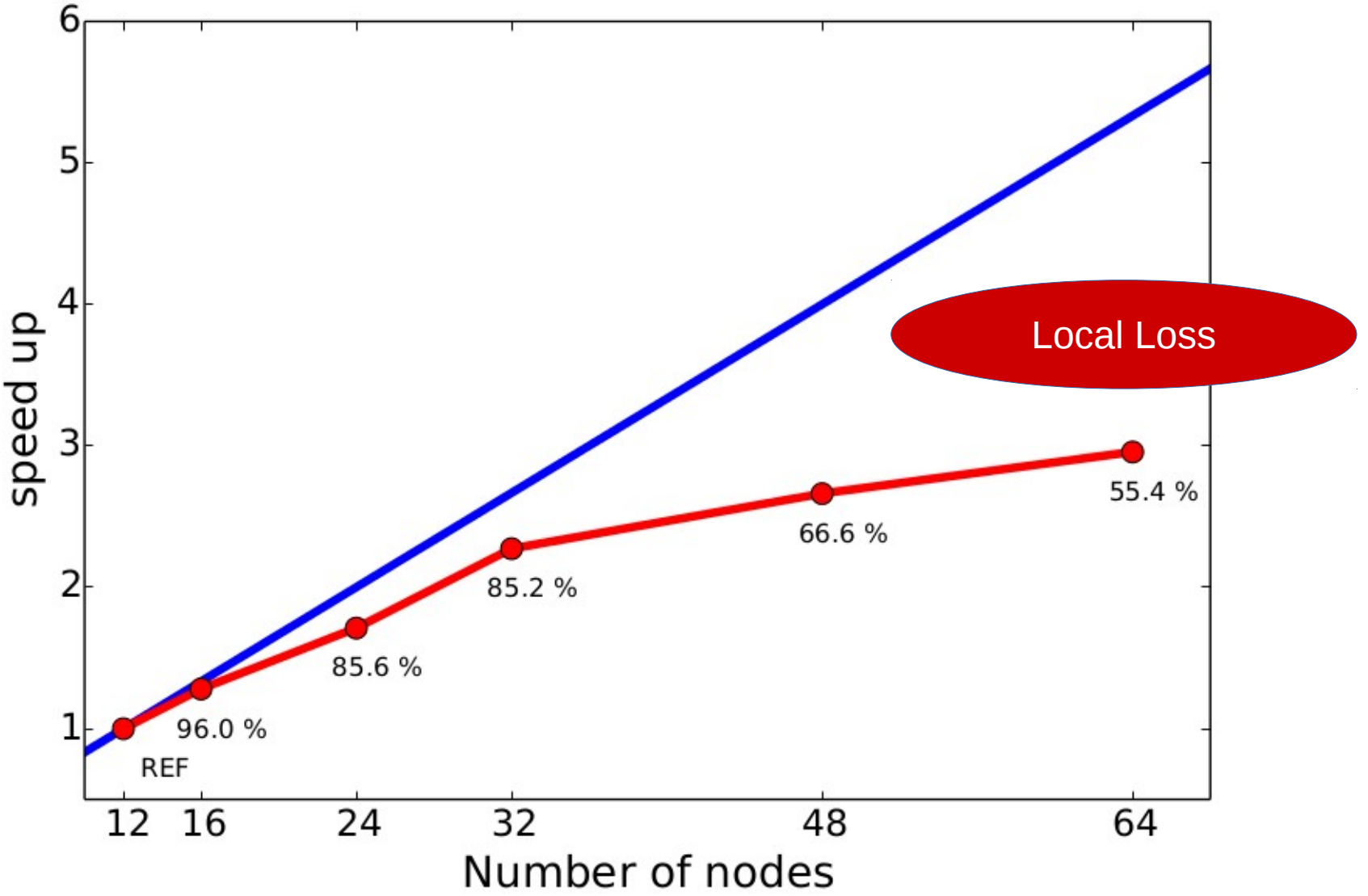
DEC parallelization



Medium grained parallelization



Medium grained parallelization



Divide-Expand-Consolidate

- Linear scaling with system size
- Massively parallel
- Maintain error control
- Strong scaling
- Weak scaling
- MPI fault tolerance ready

Divide-Expand-Consolidate

- Limited by biggest fragment
  - Correspond to a standard calculation in a subspace

A Large fragment calculation:
Number of Occupied = 256 of 4760
Number of Virtual = 1464 of 19680

1123 GB of 70202 TB

PDM can now be used: 100 nodes 10 GB pr node

DEC fragment calculations

$$\sum_{cd} t_{cidj} g_{acbd} + \sum_{cd} \sum_{ab} t_{akbl} t_{cidj} g_{kcld} + \cdots$$

$$-\frac{1}{2} \sum_{ck} t_{bkcj} g_{kiac} - \sum_{ck} t_{bkci} g_{kjac} + \cdots$$

t_{aibj}	Stored in parallel distributed memory
g_{aibj}	Stored in parallel distributed memory

DEC fragment calculations

$$\sum_{cd} t_{cidj} g_{acbd} + \sum_{cd} \sum_{ab} t_{akbl} t_{cidj} g_{kcld} + \cdots$$

$$-\frac{1}{2} \sum_{ck} t_{bkcj} g_{kiac} - \sum_{ck} t_{bkci} g_{kjac} + \cdots$$

t_{aibj} Stored in parallel distributed memory

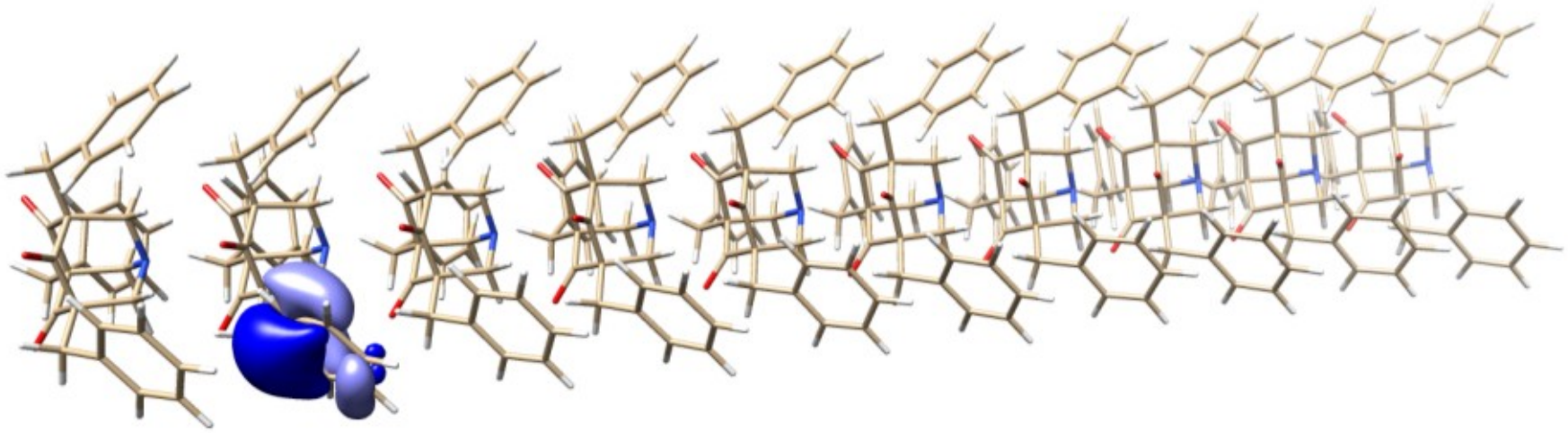
g_{aibj} Stored in parallel distributed memory

ScaTeLib: A scalable tensor library

Patrick Ettenhuber & Dmitry Liakh

DEC performance

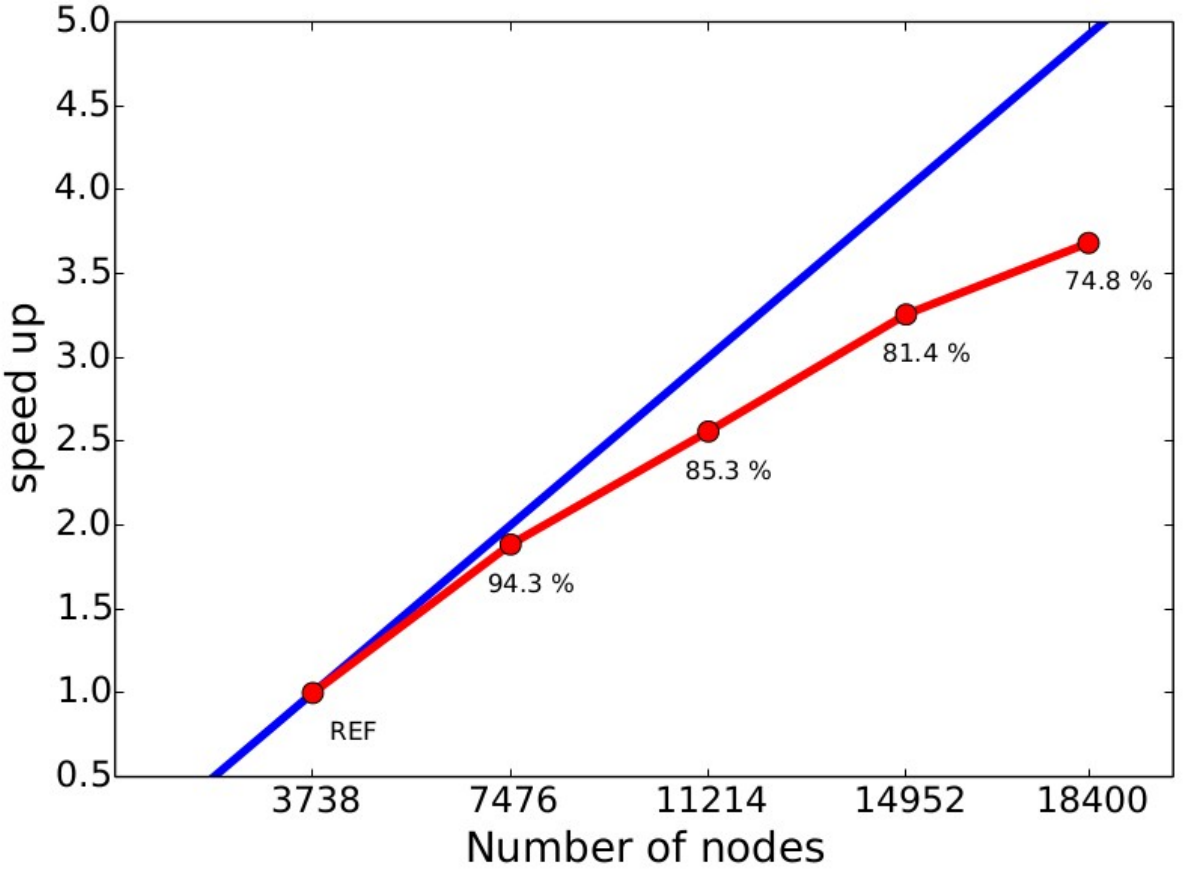
Gordon Bell submission: DEC-RIMP2



Stacks of 1-aza-adamantane-trione (AAT) supramolecular wires

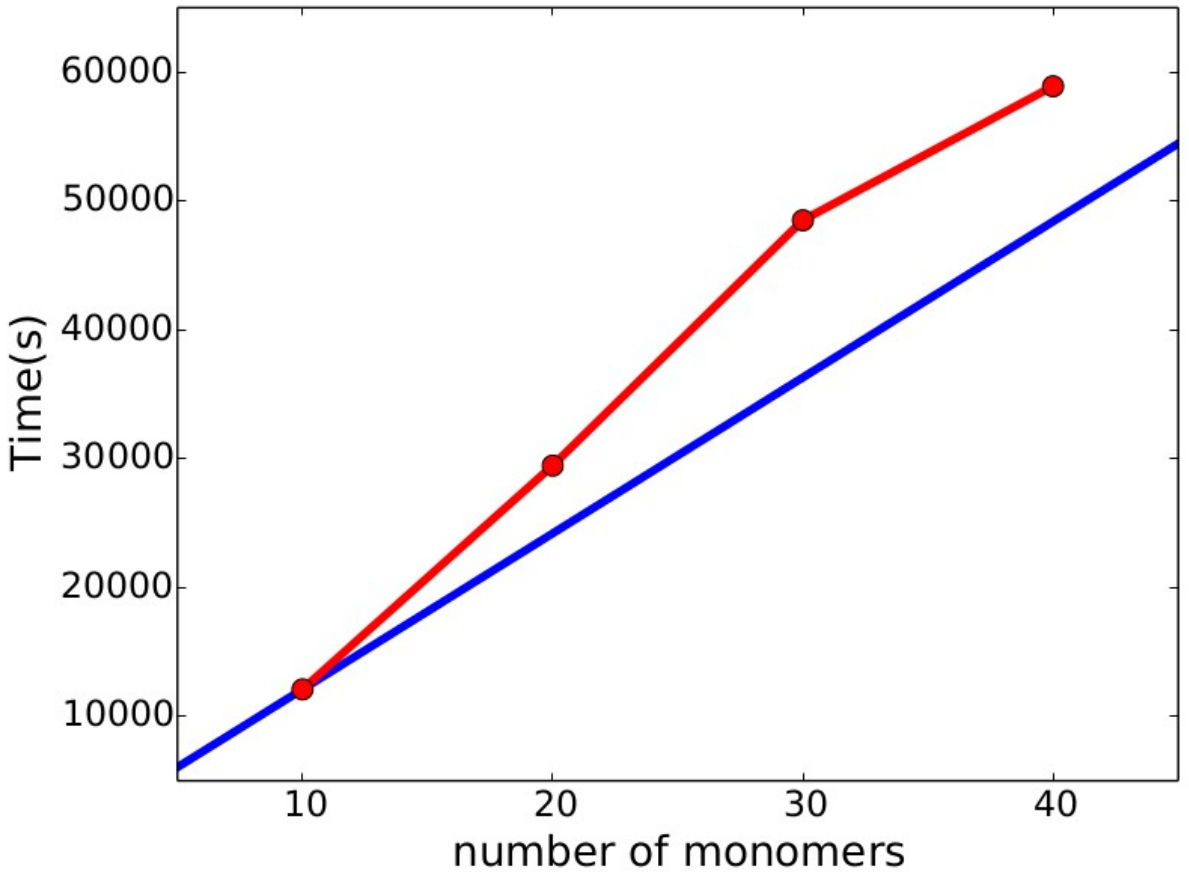
Conventional required 70202 TB data

DEC performance



Strong Scaling

DEC performance



Linear Scaling

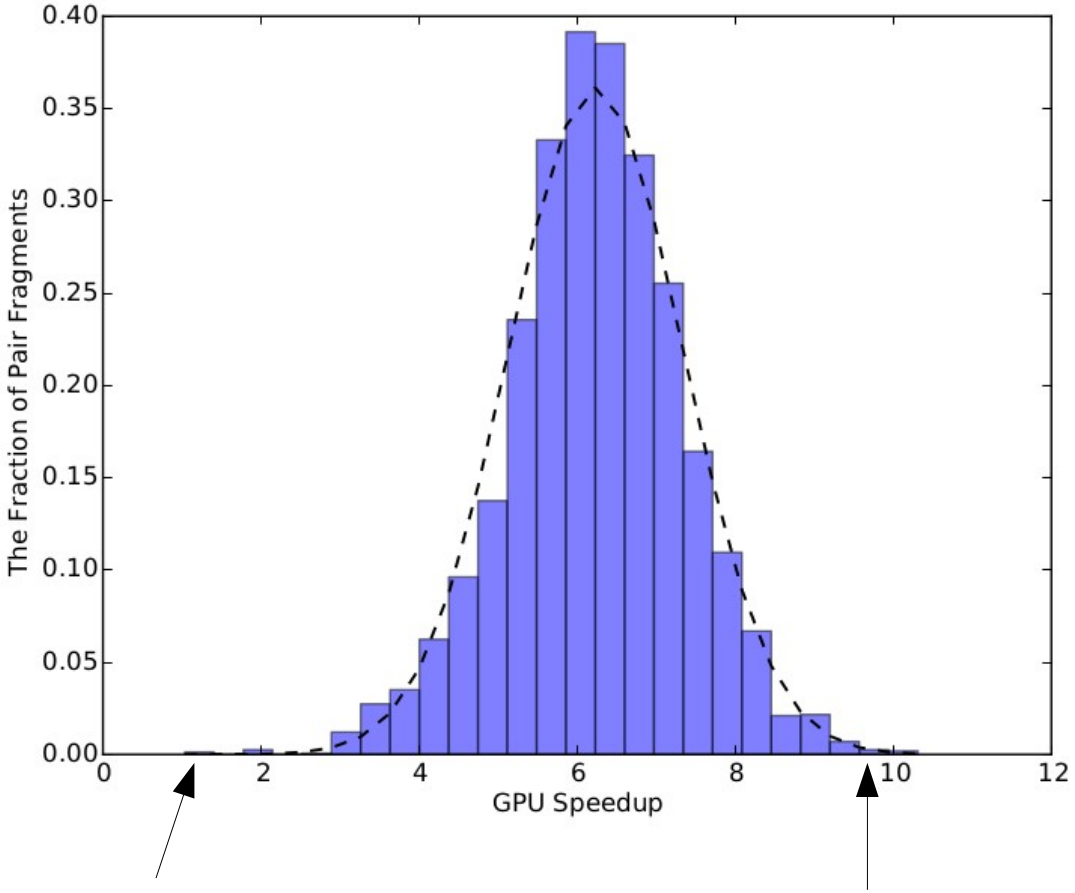
DEC performance

$$E_{\mathrm{ws}} = \frac{F_j P_i T_i}{(F_i P_j T_j)}$$

System	# nodes	EFLOPs	TTS(s)	$\mathrm{E_{ws}}$ (%)
AAT_{10}	3738	4.08	47570	
AAT_{20}	7476	9.41	60430	90.8
AAT_{30}	11214	15.0	71180	81.9
AAT_{40}	14952	19.4	68060	83.1

Weak Scaling

DEC performance



1 second

2 hours (16 nodes)

DEC fragment calculations

Challenges

- Memory requirements N^4
- Computational scaling N^6,N^7,N^8

Wish list

- PDM storage
- Parallel tensor contraction
- GPU/accelerators utilization

Summary

- OpenACC know-how in the group
- DEC code
  - Linear Scaling
  - Massively parallel
  - Strong Scaling
  - Weak Scaling
  - Partly ported to GPU

Acknowledgments

- Oak Ridge National Laboratory
  - Tjerk Straatsma, Dmitry Liakh
- NVIDIA
  - Mark Berger, Jeff Larkin
- Prof. Poul Jørgensen
  - Kasper Kristensen, Pablo Baudin, Dmytro Bykov, Janus Juul Eriksen, Patrick Ettenhuber, Filip Pawlowski, Yang Min Wang