How to Save TERABYES of memory:

on-the-fly algorithms for 3- (and 4-) body forces in many-body systems

Collaborators:

W. Erich Ormand, Lawrence Livermore

Plamen G. Krastev, Harvard Supercomputing

Hai Ah Nam, SDSU/ Oak Ridge

Collaborators-in-training:

Joshua Staker & Micah Schuster, SDSU

TRIUMF – FEB 2012

How to Save ~~TERABYTES~~ PETABYES of memory:

on-the-fly algorithms for 3- (and 4-) body forces in many-body systems

Collaborators:

W. Erich Ormand, Lawrence Livermore

Plamen G. Krastev, Harvard Supercomputing

Hai Ah Nam, SDSU/ Oak Ridge

Collaborators-in-training:

Joshua Staker & Micah Schuster, SDSU

TRIUMF – FEB 2012

The Key Issues

Sparsity/storage requirements of matrices

Redundancy of matrix elements

Factorization and reduced storage of operations

Parallel distribution of operations

What's new in BIGSTICK

SPARSITY
AND MATRIX STORAGE

A Sparse Matrix, but….

Despite sparsity, nonzero matrix elements can require TB of storage

- Typical dimensions and sparsity

Nuclide	valence space	valence Z	valence N	basis dim	sparsity (%)	
^{20}Ne	"sd"	2	2	640	10	
^{25}Mg	"sd"	4	5	44,133	0.5	
^{49}Cr	"pf"	4	5	6M	0.01	
^{56}Fe	"pf"	6	10	500M	$2x10^{-4}$	
^{12}C	N_{max}=8	6	6	600M	$4x10^{-4}$	2-body force
^{12}C	N_{max}=8	6	6	600M	$2x10^{-2}$	3-body force

A Sparse Matrix, but....

Despite sparsity, nonzero matrix elements can require TB of storage

Nuclide	Space	Basis dim	matrix store
^{56}Fe	*pf*	501 M	4.2 Tb
^{7}Li	N_{max}=12	252 M	3.6 Tb
^{7}Li	N_{max}=14	1200 M	23 Tb
^{12}C	N_{max}=6	32M	0.2 Tb
^{12}C	N_{max}=8	590M	5 Tb
^{12}C	N_{max}=10	7800M	111 Tb
^{16}O	N_{max}=6	26 M	0.14 Tb
^{16}O	N_{max}=8	990 M	9.7 Tb

A SPARSE MATRIX, BUT….

Despite sparsity, nonzero matrix elements can require TB of storage

Nuclide	Space	Basis dim	matrix store (2-body)	matrix store (3-body)
^{4}He	$N_{max}=16$	6 M	0.2 Gb	12 Tb
^{4}He	$N_{max}=20$	39 M	3 Tb	270 Tb
^{7}Li	$N_{max}=10$	43 M	0.4 Tb	176 Tb
^{12}C	$N_{max}=6$	32M	0.2 Tb	6.2 Tb
^{12}C	$N_{max}=8$	590M	5 Tb	200 Tb

Redundancy of matrix elements

A SPARSE MATRIX, BUT....

- How the Hamiltonian is represented

"occupation representation" $|\alpha\rangle = \hat{a}^+_{n_1}\hat{a}^+_{n_2}\hat{a}^+_{n_3}\ldots\hat{a}^+_{n_N}|0\rangle$

n_i	1	2	3	4	5	6	7
α=1	1	0	0	1	1	0	1
α=2	1	0	1	0	0	1	1
α=3	0	1	1	1	0	1	0

$$\hat{H} = \sum_{ij} T_{ij}\hat{a}^+_i\hat{a}_j + \frac{1}{4}\sum_{ijkl} V_{ijkl}\hat{a}^+_i\hat{a}^+_j\hat{a}_l\hat{a}_k$$

Usually, each *i* represents single-particle states with good *j*,*m*, parity

RECYCLED MATRIX ELEMENTS

Only a fraction of matrix elements are unique; **most are reused.**

Reuse of matrix elements understood through *spectator* particles.

n_i	1	2	3	4	5	6	7	8
α=1	1	1	1	0	0	0	0	1
α=2	1	1	0	1	1	0	0	0
α=3	0	1	1	0	0	1	0	1
α=4	0	1	0	1	1	1	0	0
α=5	0	0	1	0	0	1	1	1
α=6	0	0	0	1	1	1	1	0

$$\hat{a}_4^+\hat{a}_5^+\hat{a}_3\hat{a}_8|\alpha = 1\rangle = |\alpha = 2\rangle$$

$$\hat{a}_4^+\hat{a}_5^+\hat{a}_3\hat{a}_8|\alpha = 3\rangle = |\alpha = 4\rangle$$

$$\hat{a}_4^+\hat{a}_5^+\hat{a}_3\hat{a}_8|\alpha = 5\rangle = |\alpha = 6\rangle$$

All of these have the same matrix element: V_{4538}

Recycled Matrix Elements

Only a fraction of matrix elements are unique; **most are reused.**
Reuse of matrix elements understood through *spectator* particles.

# of nonzero matrix elements vs. # unique matrix elements

Nuclide	valence space	valence Z	valence N	# nonzero	# unique
^{28}Si	"sd"	6	6	26×10^6	3600
^{52}Fe	"pf"	6	6	90×10^9	21,500

Nuclide	ab initio space	basis dim	# nonzero m.e.s	# unique	avg redundancy
^{4}He	$N_{max}=16$	6M	2×10^{10}	10^9	18
^{12}C	$N_{max}=8$	600M	6×10^{11}	5×10^7	10,000

FACTORIZATION ALGORITHMS

Factorization

Reuse can be **exploited using exact factorization**
enforced through *additive/multiplicative quantum numbers*

We work in an *M*-scheme basis:

Because $\mathbf{J}^2$ and $\mathbf{J}_z$ both commute with $\mathbf{H}$, one does not need *all* basis states, but can use many-body basis restricted to the same *M*.

This is easy because *M* is an additive quantum number so it is possible for a single Slater determinant to be a state of good *M*.

(It's possible to work in a *J*-basis, e.g. OXBASH or NuShell, but each basis state is generally a complicated sum of Slater determinants).

FACTORIZATION

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

Because the M values are discrete integers or half-integers (-3, -2, -1, 0, 1, 2, … or -3/2, -1/2, +1/2, +3/2….) we can organize the basis states in discrete *sectors*

Example: 2 protons, 4 neutrons, total M = 0

$M_z(\pi) = -4$	$M_z(\upsilon) = +4$
$M_z(\pi) = -3$	$M_z(\upsilon) = +3$
$M_z(\pi) = -2$	$M_z(\upsilon) = +2$

FACTORIZATION

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

In fact, we can see an example of factorization here because all proton Slater determinants in one M-sector *must* combine with all the conjugate neutron Slater determinants

Example: 2 protons, 4 neutrons, total M = 0

$M_z(\pi) = -4$: 2 SDs $M_z(\upsilon) = +4$: 24 SDs 48 combined

$M_z(\pi) = -3$: 4 SDs $M_z(\upsilon) = +3$: 39 SDs 156 combined

$M_z(\pi) = -2$: 9 SDs $M_z(\upsilon) = +2$: 60 SDs 540 combined

FACTORIZATION

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

In fact, we can see an example of factorization here because all proton Slater determinants in one M-sector *must* combine with all the conjugate neutron Slater determinants

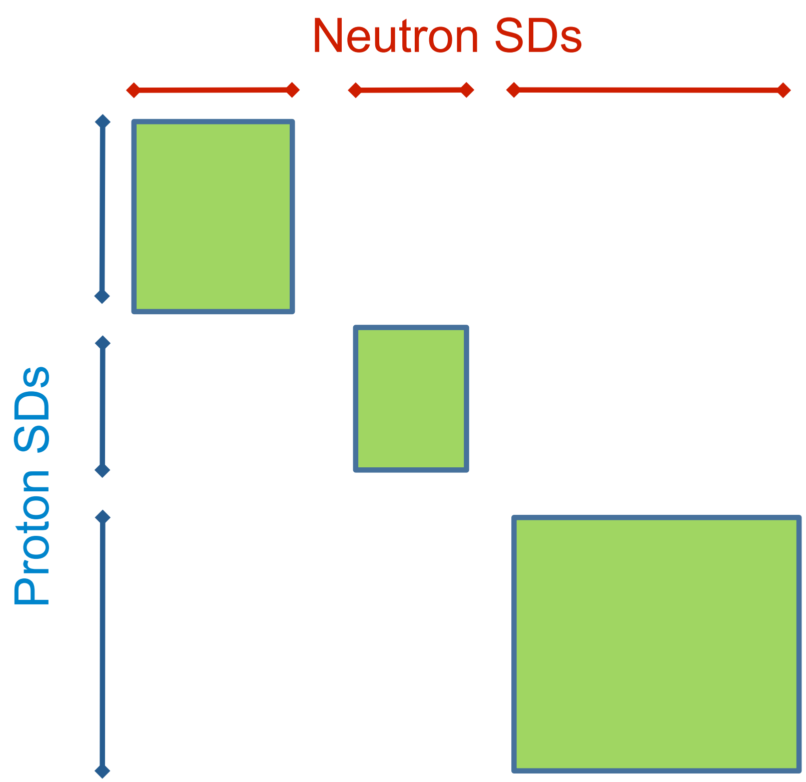
$M_z(\pi)$ = -4: 2 SDs	$M_z(\upsilon)$ = +4: 24 SDs	48 combined

$$
\begin{matrix} |\pi_1\rangle \\ |\pi_2\rangle \end{matrix} \quad \times \quad \begin{matrix} |\nu_1\rangle \\ |\nu_2\rangle \\ |\nu_3\rangle \\ |\nu_4\rangle \\ \vdots \\ |\nu_{24}\rangle \end{matrix} \quad = \quad \begin{matrix} |\pi_1\rangle|\nu_1\rangle \\ |\pi_2\rangle|\nu_1\rangle \\ |\pi_1\rangle|\nu_2\rangle \\ |\pi_2\rangle|\nu_2\rangle \\ \vdots \\ |\pi_1\rangle|\nu_{24}\rangle \\ |\pi_2\rangle|\nu_{24}\rangle \end{matrix}
$$

Factorization

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

$$|\alpha\rangle = |\alpha_p\rangle \times |\alpha_n\rangle$$



Example N = Z nuclei

Nuclide	Basis dim	# pSDs (=#nSDs)
^{20}Ne	640	66
^{24}Mg	28,503	495
^{28}Si	93,710	924
^{48}Cr	1,963,461	4895
^{52}Fe	109,954,620	38,760
^{56}Ni	1,087,455,228	125,970

FACTORIZATION

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

Factorization allows us to keep track of all basis states without writing out every one explicitly
-- we only need to write down the proton/neutron components

The same trick can be applied to matrix-vector multiply

$$\hat{H} = \hat{H}_{pp} + \hat{H}_{nn} + \hat{H}_{pn}$$

Move 2 protons; *neutrons are spectators*

Move 2 neutrons; *protons are spectators*

Move 1 proton + 1 neutron; *rest are spectators*

FACTORIZATION

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

$$\hat{H}_{pp}$$

Move 2 protons; *neutrons are spectators*

Example: 2 protons, 4 neutrons, total M = 0

$M_z(\pi) = -4$: 2 SDs	$M_z(\upsilon) = +4$: 24 SDs	48 combined

There are potentially 48 × 48 matrix elements
But for H_{pp} at most 4 × 24 are nonzero
and we only have to look up 4 matrix elements

Advantage: **we can store 98 matrix elements as 4 matrix elements** and avoid 2000+ zero matrix elements.

Factorization

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

$M_z(\pi)$ = -4: 2 SDs

$M_z(\upsilon)$ = +4: 24 SDs

48 combined

$$|\pi_1\rangle \qquad |\pi_2\rangle$$

$$H_{pp} = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix}$$

$$|\nu_1\rangle \quad |\nu_2\rangle \quad |\nu_3\rangle \quad |\nu_4\rangle \quad \vdots \quad |\nu_{24}\rangle$$

Advantage: **we can store 98 matrix elements as 4 matrix elements** and avoid 2000+ zero matrix elements.

FACTORIZATION

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

$M_z(\pi)$ = -4: 2 SDs

$M_z(\upsilon)$ = +4: 24 SDs

48 combined

$$|\pi_1\rangle \qquad |\pi_2\rangle$$

$$H_{pp} = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix}$$

$$|\nu_1\rangle \quad |\nu_2\rangle \quad |\nu_3\rangle \quad |\nu_4\rangle \quad \vdots \quad |\nu_{24}\rangle$$

$$H_{pp}|\pi_1\rangle|\nu_1\rangle = H_{11}|\pi_1\rangle|\nu_1\rangle + H_{12}|\pi_2\rangle|\nu_1\rangle$$
$$H_{pp}|\pi_2\rangle|\nu_1\rangle = H_{12}|\pi_1\rangle|\nu_1\rangle + H_{22}|\pi_2\rangle|\nu_1\rangle$$
$$H_{pp}|\pi_1\rangle|\nu_2\rangle = H_{11}|\pi_1\rangle|\nu_2\rangle + H_{12}|\pi_2\rangle|\nu_2\rangle$$
$$H_{pp}|\pi_2\rangle|\nu_2\rangle = H_{12}|\pi_1\rangle|\nu_2\rangle + H_{22}|\pi_2\rangle|\nu_2\rangle$$
$$\vdots$$
$$H_{pp}|\pi_1\rangle|\nu_{24}\rangle = H_{11}|\pi_1\rangle|\nu_{24}\rangle + H_{12}|\pi_2\rangle|\nu_{24}\rangle$$
$$H_{pp}|\pi_2\rangle|\nu_{24}\rangle = H_{12}|\pi_1\rangle|\nu_{24}\rangle + H_{22}|\pi_2\rangle|\nu_{24}\rangle$$

Advantage: **we can store 98 matrix elements as 4 matrix elements** and avoid 2000+ zero matrix elements.

Factorization

Reuse can be **exploited using exact factorization** enforced through *additive/multiplicative quantum numbers*

Comparison of nonzero matrix storage with factorization

Nuclide	Space	Basis dim	matrix store	factorization
^{7}Li	N_{max}=12	252 M	3600 Gb	96 Gb
^{7}Li	N_{max}=14	1200 M	23 Tb	624 Gb
^{12}C	N_{max}=6	32M	196 Gb	3.3 Gb
^{12}C	N_{max}=8	590M	5000 Gb	65 Gb
^{12}C	N_{max}=10	7800M	111 Tb	1.4 Tb
^{16}O	N_{max}=6	26 M	142 Gb	3.0 Gb
^{16}O	N_{max}=8	990 M	9700 Gb	130 Gb

FACTORIZATION

Comparison of nonzero matrix storage with factorization

^{4}He

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{max}=14	2M	46 Gb	1.2 Gb	2 Tb	16 Gb
N_{max}=16	6M	200 Gb	4 Gb	12 Tb	60 Gb
N_{max}=18	16M	820 Gb	11 Gb	60 Tb	190 Gb
N_{max}=20	39M	3 Tb	29 Gb	270 Tb	600 Gb
N_{max}=22	86M	9 Tb	70 Gb	1.1 Pb	1.4 Tb

FACTORIZATION

Comparison of nonzero matrix storage with factorization

^{4}He

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
$N_{shell}=8$	29 M	1.4 Tb	0.6 Gb	120 Tb	11 Gb
$N_{shell}=9$	93 M	8 Tb	1.7 Gb	870 Tb	40 Gb
$N_{shell}=10$	270 M	36 Tb	5 Gb	5 Pb	120 Gb
$N_{shell}=11$	700 M	150 Tb	12 Gb	28 Pb	350 Gb
$N_{shell}=12$	1.7 G	500 Tb	27 Gb	130 Pb	900 Gb
$N_{shell}=13$	4 G	1.7 Pb	60 Gb	500 Pb	2 Tb

FACTORIZATION

Comparison of nonzero matrix storage with factorization

^{7}Li

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{max}=8	6 M	36 Gb	1.5 Gb	1 Tb	26 Gb
N_{max}=10	43 M	430 Gb	10 Gb	170 Tb	250 Gb
N_{max}=12	250 M	4 Tb	60 Gb		

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{shell}=3	0.4 M	0.8 Gb	6 Mb	10 Gb	44 Mb
N_{shell}=4	45 M	330 Gb	0.3 Gb	9 Tb	4 Gb
N_{shell}=5	2 G	38 Tb	16 Gb	2 Pb	140 Gb
N_{shell}=6	50 G	2 Pb	87 Gb	170 Pb	3 Tb

FACTORIZATION

Comparison of nonzero matrix storage with factorization

^{9}Be

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{max}=6	5 M	22 Gb	1 Gb	0.6 Tb	12 Gb
N_{max}=8	63 M	460 Gb	9 Gb	17 Tb	200 Gb
N_{max}=10	570 M	7 Tb	70 Gb		

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{shell}=3	4 M	15 Gb	30 Mb	240 Gb	240 Mb
N_{shell}=4	3 G	30 Tb	3 Gb	1 Pb	50 Gb
N_{shell}=5	400 G	12 Pb	130 Gb	800 Pb	3.6 Tb

FACTORIZATION

Comparison of nonzero matrix storage with factorization

^{10}B

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{max}=6	12 M	60 Gb	1.3 Gb	1.6 Tb	22 Gb
N_{max}=8	165 M	1.3 Tb	16 Gb	52 Tb	360 Gb

FACTORIZATION

Comparison of nonzero matrix storage with factorization

^{12}C

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{max}=6	32 M	170 Gb	3 Gb	5 Tb	60 Gb
N_{max}=8	590 M	5 Tb	45 Gb	200 Tb	1 Tb
N_{max}=10	8 G	100 Tb	440 Gb		

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
N_{shell}=3	82 M	400 Gb	0.1 Gb	9 Tb	1.5 Gb
N_{shell}=4	600 G	10 Pb	43 Gb	580 Tb	0.9 Tb

Factorization

Comparison of nonzero matrix storage with factorization

^{16}O

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
$N_{max}=4$	0.3 M	1 Gb	70 Mb	17 Gb	0.7 Gb
$N_{max}=6$	26 M	140 Gb	3 Gb	4 Tb	53 Gb
$N_{max}=8$	1 G	8.6 Tb	70 Gb		

Space	Basis dim	matrix store (2-body)	factorization (2-body)	matrix store (3-body)	factorization (3-body)
$N_{shell}=3$	800 M	6 Tb	0.7 Gb	140 Tb	7.5 Gb

Drawbacks of factorization/on-the-fly algorithms:

Much more complicated to code up (even matrix storage is not trivial)

Less flexible in basis—for example, importance truncation much harder (if even possible)

4-body is in principle straightforward

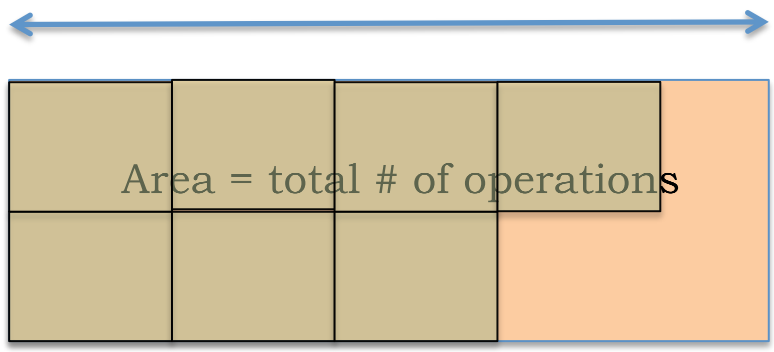
Experience in going from 2-body to 3-body shows most difficult part is correctly matching indices of input interaction to internal representation (+ induced phases etc) – useful to have *small* cases with known solutions for debugging

Parallel Implementation

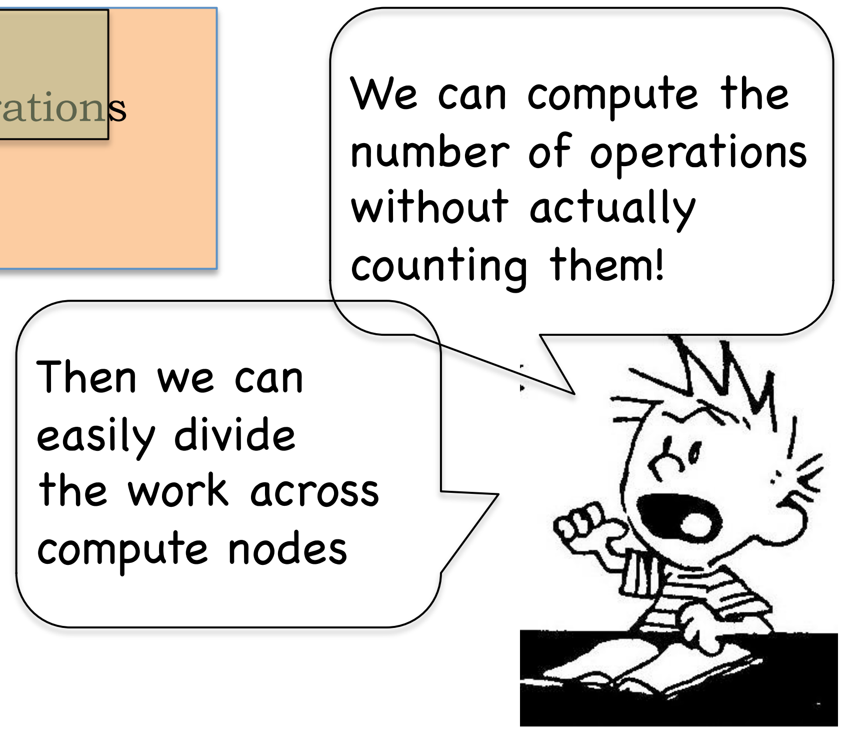
Parallel Implementation

Factorization makes it easier to compute workload
and distribute across multiple nodes

length of sides =
information to be stored

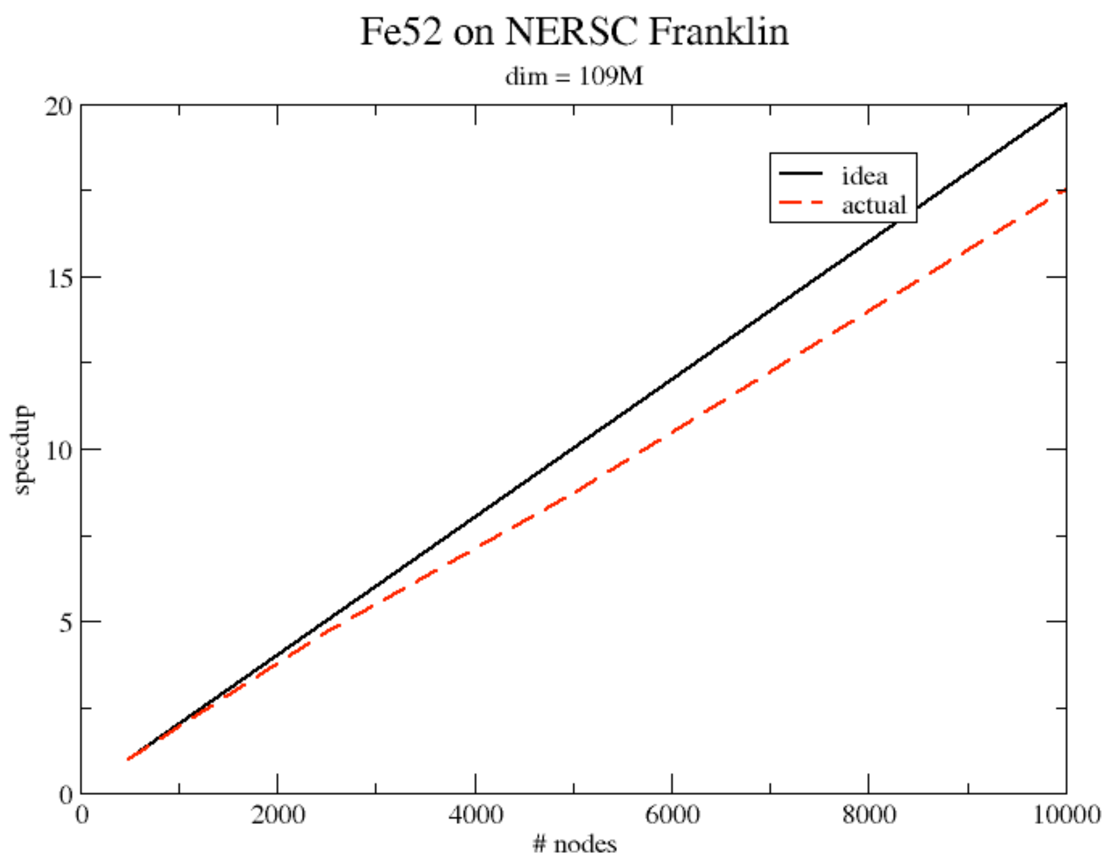


length of
sides =
information
to be stored



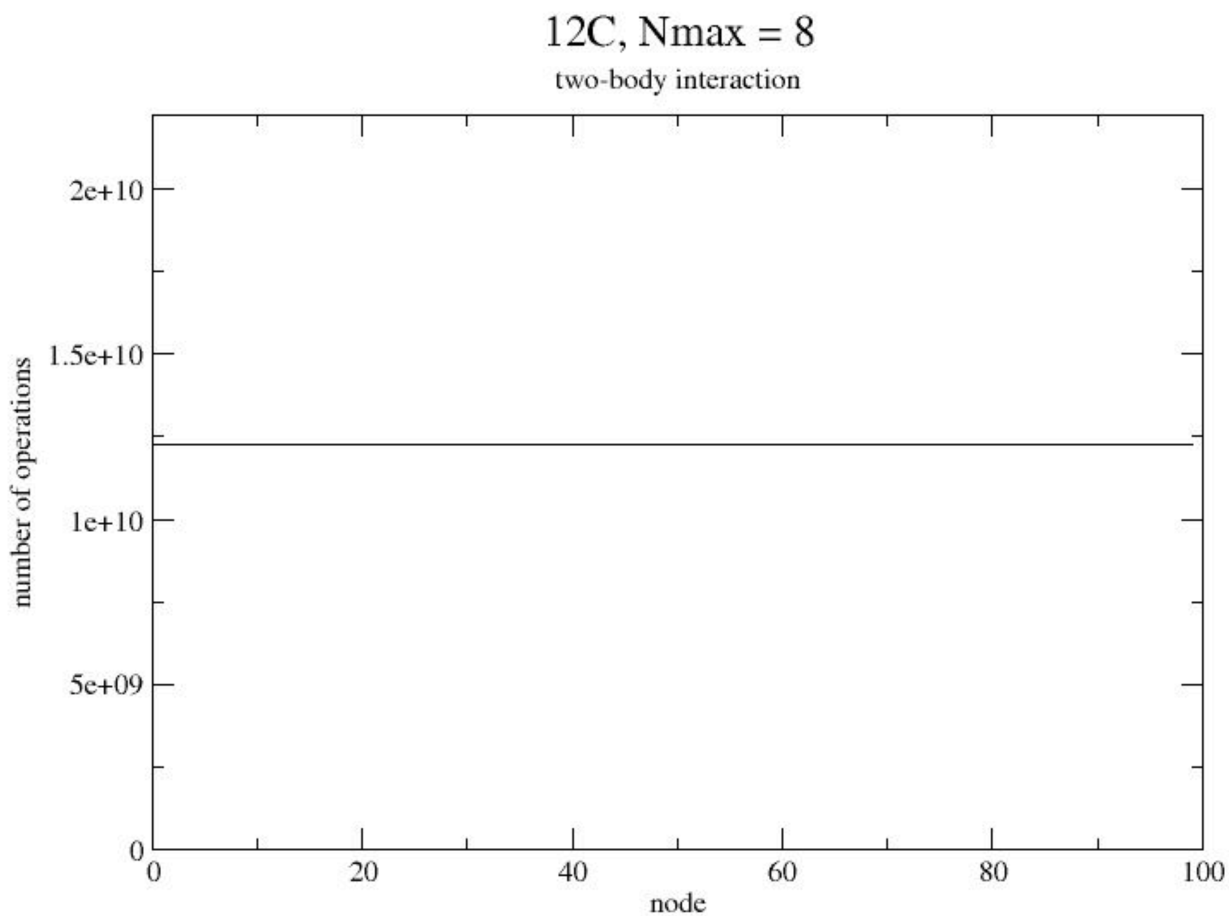
Parallel Implementation

Factorization makes it easier to compute workload and distribute across multiple nodes



PARALLEL IMPLEMENTATION

Factorization makes it easier to compute workload and distribute across multiple nodes



The BIGSTICK code

THE BIGSTICK CODE

Many-fermion code: 2nd generation after REDSTICK code
(started in *Baton Rouge, La.*)

Arbitrary single-particle radial waveforms
Allows local or nonlocal two-body interaction
Applies to both nuclear and atomic cases

Runs on both desktop and parallel machines
--can run at least dimension 100M+ on desktop
(20 Lanczos iterations in 300 CPU minutes)

20-30k lines of codes
Fortran 90 + MPI + OpenMP
Partially funded by SciDAC
Plans to run on 50,000-100,000 compute nodes
Plans to publish code late 2012

The BIGSTICK code

What's new since last year:

Full 3-body capability

Improved efficiencies in memory usage

OpenMP parallelization (WEO)

"2nd generation" MPI parallelization in progress

Looking for purveyors of 3-body interactions to partner with ! (Also, 4-body...)