COMPUTATIONAL CHALLENGES FOR 3-BODY FORCES IN THE SHELL MODEL

Hai Ah Nam

San Diego State University

Computational Science, PhD Student

Collaboration with: Calvin Johnson, SDSU
W. Erich Ormand, LLNL

This work performed under the auspices of the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

The Shell Model

- **Typical eigenvalue problem**

$$H\Psi_i = E_i \Psi_i$$

- **Construct many-body basis states $|\phi_i\rangle$ so that**

$$\Psi_i = \sum_n C_{in}\phi_n$$

- **Calculate Hamiltonian matrix elements**

$$H_{ij} = \langle \phi_j | H | \phi_i \rangle$$

  - Diagonalize to obtain eigenvalues & eigenvectors

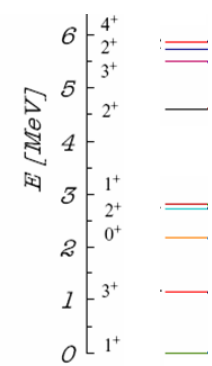
$$H = \sum_i \varepsilon_i a_i^+ a_i + \frac{1}{4}\sum_{ijkl} V_{ijkl} a_i^+ a_j^+ a_l a_k$$

$$H = \sum_i \varepsilon_i a_i^+ a_i + \sum_{ijklmn} V_{ijklmn} a_i^+ a_j^+ a_k^+ a_n a_m a_l$$

$$\phi = \frac{1}{\sqrt{A!}} \begin{vmatrix} \phi_i(\mathbf{r}_1) & \phi_i(\mathbf{r}_2) & \cdots & \phi_i(\mathbf{r}_A) \\ \phi_j(\mathbf{r}_1) & \phi_j(\mathbf{r}_2) & & \phi_j(\mathbf{r}_A) \\ & & \ddots & \vdots \\ \phi_l(\mathbf{r}_1) & \phi_l(\mathbf{r}_2) & \cdots & \phi_l(\mathbf{r}_A) \end{vmatrix}$$

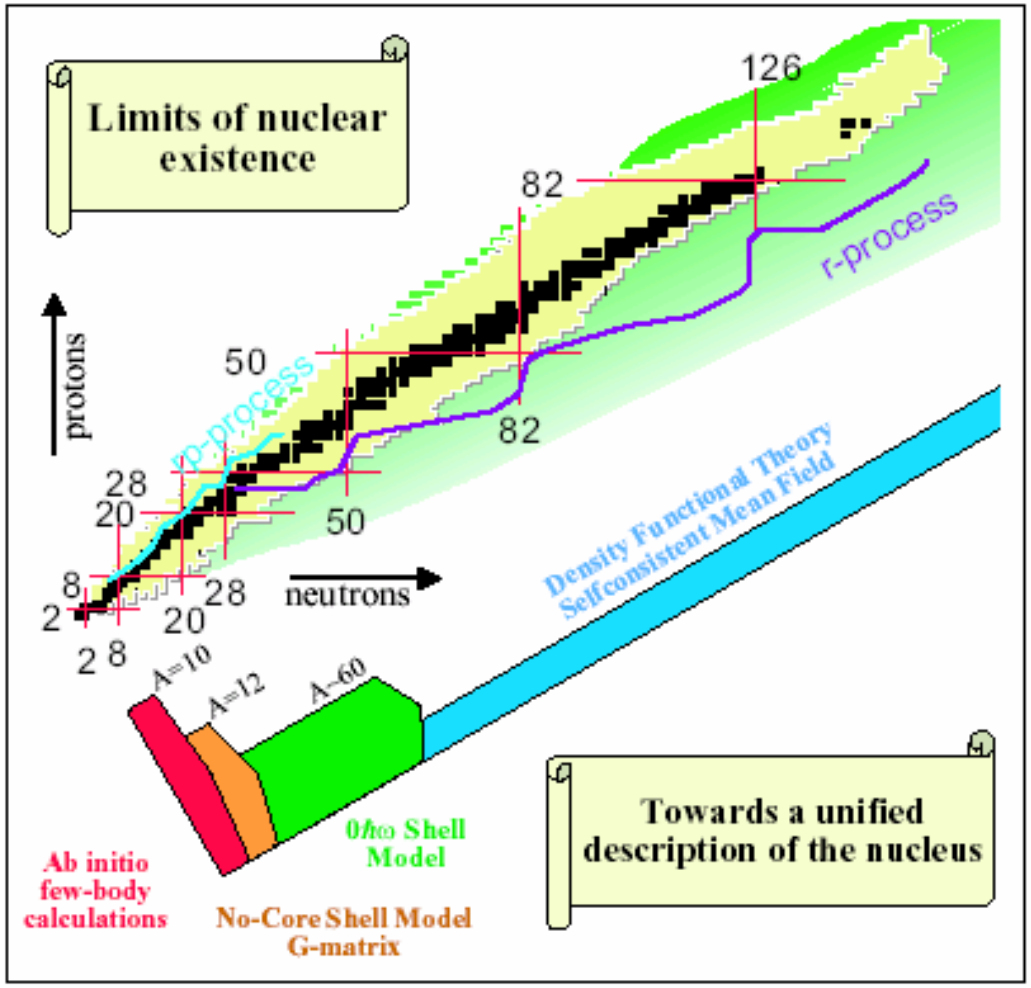
$$= a_i^+ \ldots a_j^+ a_l^+ |0\rangle$$

$$\begin{pmatrix} H_{11} & H_{12} & \cdots & H_{1N} \\ \vdots & H_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ H_{N1} & & & H_{NN} \end{pmatrix} \longrightarrow$$



Shell Model Codes

- **Oak Ridge (1969)**
  - Coefficients of Fractional parentage
- **Glasgow (1977)**
  - Good Jz (M-scheme)
  - J restored in diagonalization
- **OXBASH (1985)**
  - J-projected M-scheme
  - Smaller matrices
- **RITSSCHIL (1985)**
  - CFP
- **DUSM (1989)**
  - Permutation groups
- **ANTOINE (1991 & 1999)**
  - M-scheme
  - Apply matrix on-the-fly
  - Large dimensions
- **NATHAN (1998)**
  - J-projected similar to ANTOINE
  - "Hybrid" M-scheme-CFP code
- **REDSTICK (now)**
  - Based on ANTOINE papers
  - M-scheme
  - Three-body interactions
- **CMICHSM & MFD (now)**



Effects of 3 Body Interactions

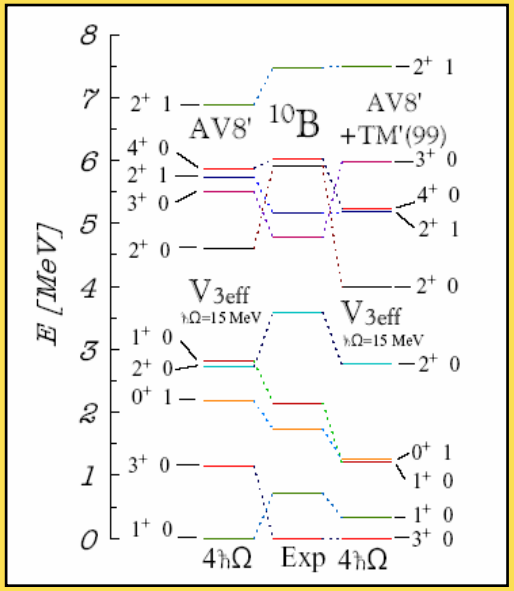
- **Increase total binding energy**
- **Increase spin-orbit splitting**
  - Improve low-lying excitation spectra
    (correct ground state spin)
- **Spin-observables**
  - Magnetic moment
  - Gamow-Teller transition strengths

BETTER AGREEMENT WITH EXPERIMENT!!

^{6}Li basis space	Exp -	AV8′+TM′(99) $6\hbar\Omega$	AV8′ $6\hbar\Omega$
$\lvert E_{gs}\rvert(1^+0)$	31.995	31.036	28.406

(2003) Investigations of ^{6,7}Li, ^{6}He, ^{7,8,10}Be, ^{10,11,12}B, ^{12}N, ^{10,11,12,13}C

Source: Navratil and Ormand, Phys. Rev. C 68, 034305



^{11}B→^{11}C basis space	Exp -	AV8′+TM′(99) $4\hbar\Omega$	AV8′ $4\hbar\Omega$
$B(GT;\frac{3}{2}_1\frac{1}{2}\rightarrow\frac{3}{2}_1\frac{1}{2})$	0.345	0.315	0.765
$B(GT;\frac{3}{2}_1\frac{1}{2}\rightarrow\frac{1}{2}_1\frac{1}{2})$	0.399	0.591	0.909
$B(GT;\frac{3}{2}_1\frac{1}{2}\rightarrow\frac{5}{2}_1\frac{1}{2})$	0.961^a	0.517	0.353
$B(GT;\frac{3}{2}_1\frac{1}{2}\rightarrow\frac{3}{2}_2\frac{1}{2})$	0.961^a	0.741	0.531
$B(GT;\frac{3}{2}_1\frac{1}{2}\rightarrow\frac{5}{2}_2\frac{1}{2})$	0.444^b	0.625	0.197

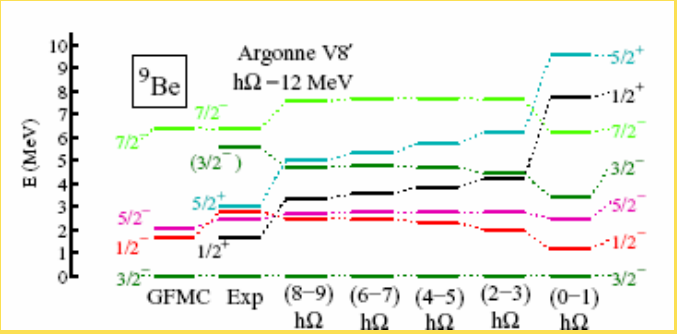
Current Investigations

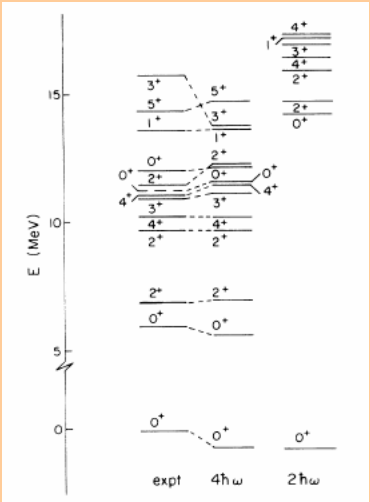
- **Higher p-shell, low sd-shell nuclei**
  - ^{9}Be, ^{15}O, ^{16}O, ^{17}O
- **2-body results**

^{9}Be	Exp	INOY	NCSM CDB2k	NCSM N^{3}LO	NCSM AV8′	GFMC AV8′
$E_{gs}\left(\frac{3}{2}^-_1\ \frac{1}{2}\right)$	-58.16	-56.05	-51.16	-50.47	-50.20	-49.9(2)
$E\left(\frac{1}{2}^+_1\ \frac{1}{2}\right)$	-56.48	-50.95	-47.81	-47.57	-46.84	—

^{9}Be

Source: Forssen, Navratil, Ormand, Caurier Physical Review C, vol. 71,Issue 4, id. 044312 (2005)





^{16}O

4hw

Source: Haxton & Johnson (1990) Phys. Rev. Let.

^{15}O, ^{16}O, ^{17}O

Nucleus	Interaction N^{3}LO	CD-Bonn	V_{1s}	Expt
^{15}O	6.158	6.643	4.789	7.464
^{15}N	6.339	6.810	4.957	7.699
^{16}O	6.951	7.444	5.469	7.976
^{17}O	6.722	7.201	5.214	7.751
^{17}F	6.559	7.048	5.059	7.542

Source: Wloch et. Al. (2005) *J. Phys. G: Nucl. Part. Phys.* **31** S1291-S1299

The three nucleon interaction plays a critical role in determining the structure of nuclei BUT is computationally challenging

Primary Operation

- **Lanczos (iterative method)**

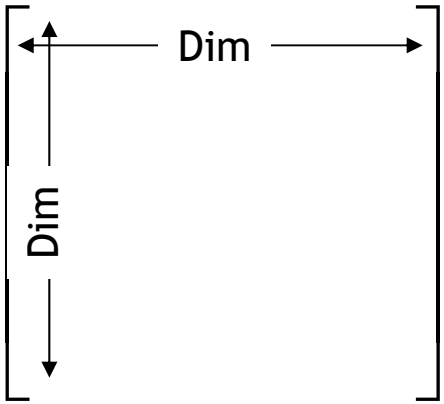
$$
\begin{aligned}
\hat{H}\mathbf{v}_1 &= \alpha_1\mathbf{v}_1 + \beta_1\mathbf{v}_2 \\
\hat{H}\mathbf{v}_2 &= \beta_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \beta_2\mathbf{v}_3 \\
\hat{H}\mathbf{v}_3 &= \beta_2\mathbf{v}_2 + \alpha_3\mathbf{v}_3 + \beta_3\mathbf{v}_4 \\
\hat{H}\mathbf{v}_4 &= \beta_3\mathbf{v}_3 + \alpha_4\mathbf{v}_4 + \beta_4\mathbf{v}_5
\end{aligned}
$$

  - Ideal for solving a large sparse matrix
  - ~ 100-200 iterations for the lowest 10 eigenvalues
  - Matrix-vector multiplication, vector dot products for α's, β's
  - Memory and run-time bottleneck

Why is it so difficult?

- **Large dimensions of the Hamiltonian matrix**
  - grows dramatically with # of particles & valence space

$$Dim \approx \binom{N^p_{sps}}{n^p}\binom{N^n_{sps}}{n^n}$$ **e.g. ^{60}Zn (fp – shell)** $$\binom{20}{10}\binom{20}{10} = 3.4\times10^{10}$$



- **Large SPARSE matrix**
  - Only store non-zero matrix elements

Nuclide	space	N_{val}	Z_{val}	Dim basis	Sparsity (%)
^{20}Ne	sd	2	2	640	13.0
^{24}Mg	sd	4	4	28,503	0.74
^{28}Si	sd	6	6	93,710	0.34
^{46}V	pf	3	3	121,440	0.36
^{48}Cr	pf	4	4	1,963,461	0.04

Example

Dim = 10^8, Sparsity = 0.005%

# of m.e. = 10^8 X 10^8 X 0.005% =

[5.0 x 10^{11} matrix elements]

Single precision - real(4)

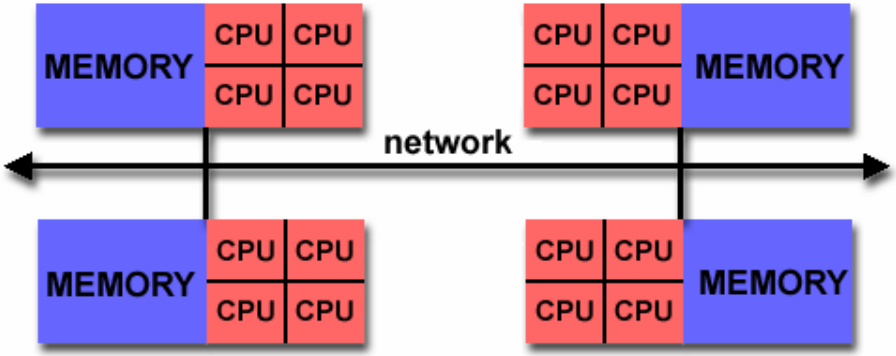
= 2 TB to store (2,000 GB)

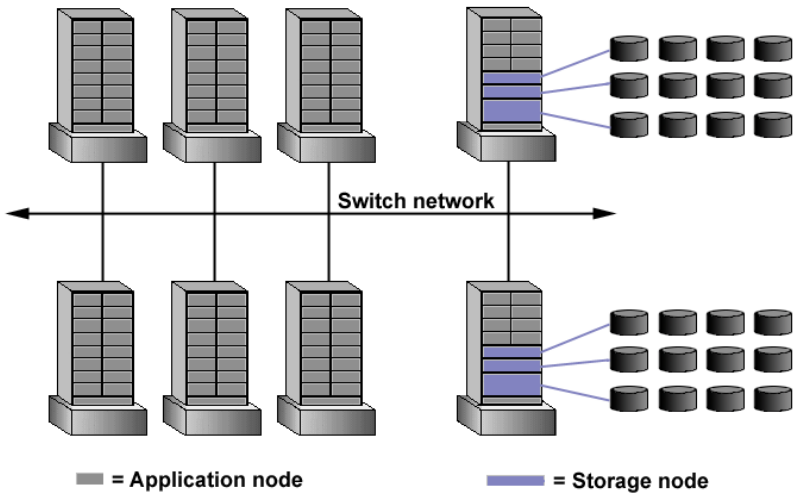
Dim = 10^9

= 200 TB (200,000 GB)

Limits of HPC Resources

- **Shared & Distributed Computing Environment**
- **Memory (RAM) ~ 2 - 4 GB/proc**
  - **Thunder** - Linux: [1024 nodes / 4096 CPUs] 4 CPUs/node w/ 8 GB shared memory (8192 GB total)
  - **uP** - IBM: [108 nodes / 864 CPUs] 8CPUs/node w/ 32 GB shared memory (3456 GB total)
- **Run-time ~ 12 hours**
  (DAT time ~ 48 hours)
- **Parallel File System (Disk)**
  - Thunder: 338 TB
  - uP: 130 TB
  - Not all available to 1 user (small fraction)





More difficulties with 3-Body Forces

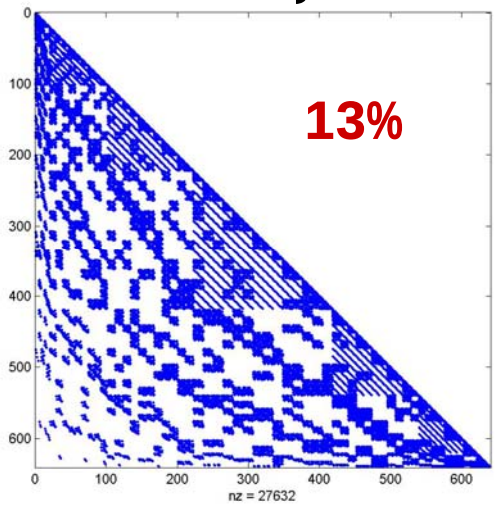
- **Basis dimensions are the same (2 → 3 body) but H is less sparse**
  - $H_{ij}=H_{pp}+H_{nn}+H_{pn}$ (2 body)
  - $H_{ijk} = H_{ppp} + H_{nnn} + H_{ppn} + H_{nnp}$ (3 body)
  - → More non-zero matrix elements
  - MORE MEMORY INTENSIVE
- **Increase in run-time**
  - **^{10}B, 4 $\hbar\Omega$; Basis Dim = *581,740***
    - 2-body has ~ 145 × 10^6 non-zero elements ~ 1- 2 CPU-hr for lowest ten states
    - *3-body has* ~ 2.2 × 10^9 non-zero elements > 200 CPU-hr
- **As if it wasn't already challenging!!**

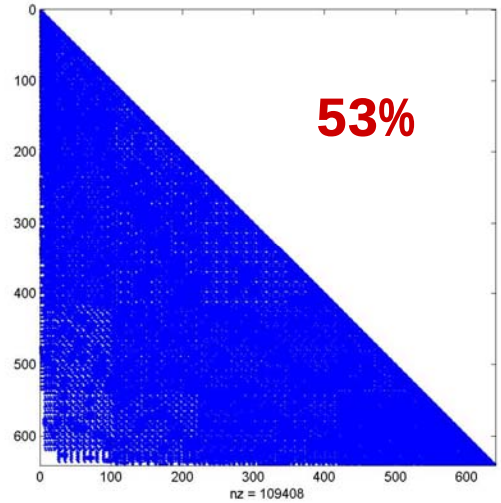
"One of us is in serious trouble!"

Sparsity Increases with 3 Body Forces

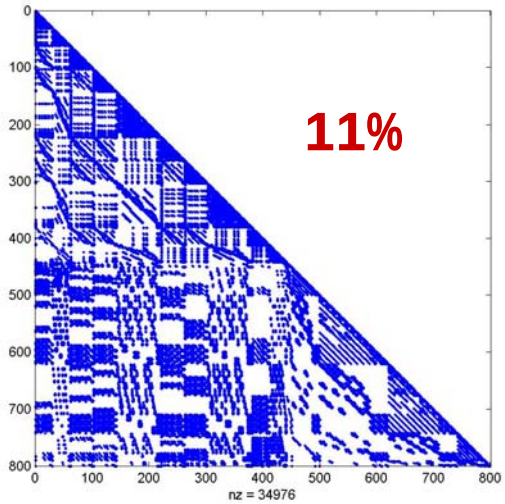
2 Body | 3 Body

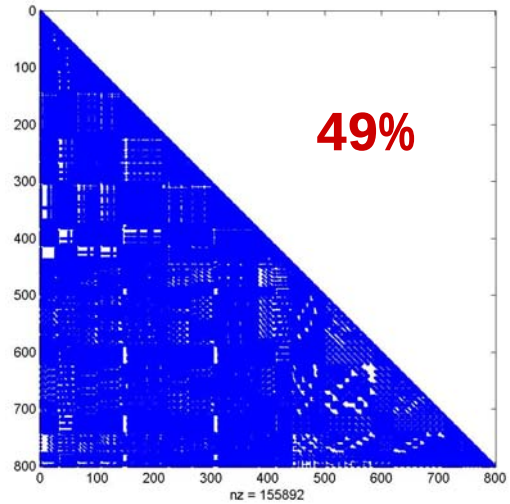
^{20}Ne
sd-shell
640 basis states





^{6}Li
2hw
800 basis states





Memory Limitation Solutions

- **Store matrix elements on disk**
  - Requires much disk space, fast i/o.
  - Disk access is ~1000 times slower than RAM access
  - (e.g., OXBASH, Glasgow-Los Alamos, CMICHSM)
- **Store matrix elements in RAM**
  - Limited by # of nodes available.
  - 3,000 processor @ 2GB = 6,000 GB RAM
  - (e.g. MFD)
- **On-the-fly: Recompute the many-body matrix elements**
  - Re-compute on each iteration from the two- (and three-) body matrix elements
  - **Efficient if you only compute non-zero matrix elements – NEED TO KNOW WHICH ARE NON-ZERO!!!**
  - (e.g. ANTOINE, REDSTICK)

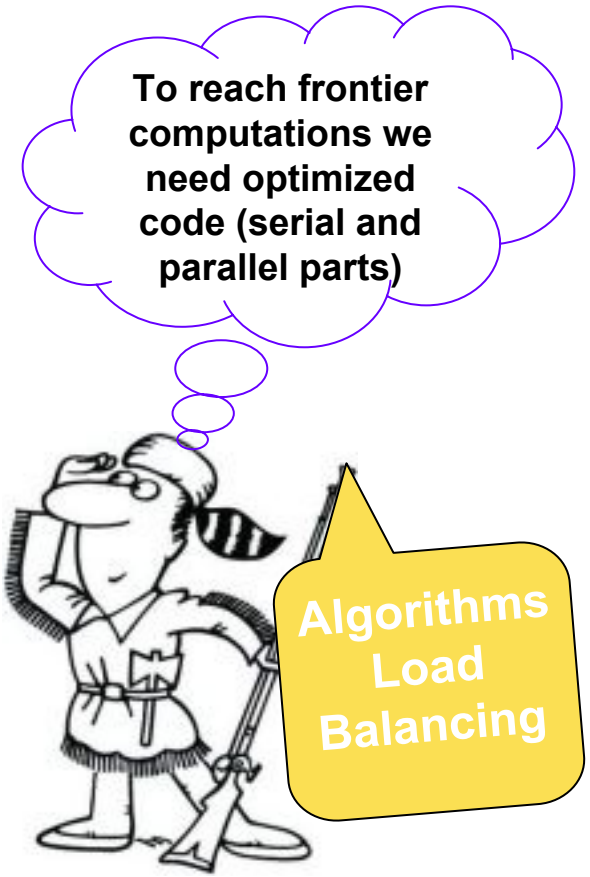
Dim = 10^8, Sparsity = 0.05%
= 2 TB (2,000 GB)

Dim = 10^9
= 200 TB (200,000 GB)

REDSTICK

- **Shell Model Code**
  - On-the-fly construction of Hamiltonian matrix
  - Created by W. Erich Ormand & Calvin W. Johnson
  - Fortran 90 & MPI
  - 2-body version
    - 65+ subroutines
    - 16,000+ lines of code
  - 3-body version
  - Release spring 2008
- **Needs Improvements**
  - Run-time
  - Current structure will have memory issues outside of lanczos bottleneck

Dim > 10^8



REDSTICK Development

- **2004 – 2005**
  - Huh? What's REDSTICK?
- **2006 – 2007**
  - New "Jump" algorithm applied to various subroutines in 2 body version
  - Improves run-time performance
- **Summer 2007**
  - New 3-body code with similar jump type algorithm
- **In Progress**
  - Analysis of parallelization schemes
  - Implementation of parallelization schemes to 3 body code
  - Improves memory utilization

Good Coding Sense

- **Avoid expensive mathematical operations**

Operation	Min Cycles per iteration (L1 Cache)
x(i) = y(i)	1.7
x(i) = x(i) + y(i)	1.7
x(i) = x(i) + s*y(i)	1.7
x(i) = 1/y(i)	15.1
x(i) = sqrt(y(i))	18.1

```
do i = 1, N
  do j = 1, N
    y(j,i) = x(j,i) / r(i)
  end do
end do
```

```
do i = 1, N
  oner = 1.0d0/r(i)
  do j = 1, N
    y(j,i) = x(j,i) * oner
  end do
end do
```

- **Avoid branching within inner loops**

```
do i=1,n
  if (r < 1.0e-16) then
  a(i)=0.0; b(i)=0.0; c(i)=0.0
  else
    a(i)=x(i)/r
    b(i)=y(i)/r
    c(i)=z(i)/r
  end if
end do
```

→

```
if (r < 1.0e-16) then
  do i=1,n
   a(i)=0.0; b(i)=0.0; c(i)=0.0
  end do
else
  do i=1,n
    a(i)=x(i)/r
    b(i)=y(i)/r
    c(i)=z(i)/r
  end do
end if
```

Algorithm Improvement: Compare vs. Jump

- **Efficiently establish supporting arrays to determine non-zero matrix elements**
  - E.g. $\hat{H}^{pn} = \sum_{ijkl} V_{ijkl}^{pn} \pi_i^+ \pi_j \nu_k^+ \nu_l$

$$H_{pn} = \sum_{ijkl} \underbrace{\left\langle \Phi_f^p \middle| \pi_i^+ \pi_j \middle| \Phi_i^p \right\rangle}_{\text{One body jump}} \underbrace{\left\langle \Phi_f^n \middle| \nu_k^+ \nu_l \middle| \Phi_i^n \right\rangle}_{\text{One body jump}} V_{ijkl}^{pn}$$

If *i* and *f* connect by a one-body operator, then m.e. ≠ 0

- **"Jump" Algorithm**

$$\left| \Phi_i \right\rangle \longrightarrow \left\langle \Phi_f \right|$$

$$\pi_a^+ \pi_b \; ?$$

COMPARE initial to final states to see if they're connected by a one-body operator

$$\left| \Phi_i \right\rangle \longrightarrow \pi_a^+ \pi_b \left| \Phi_i \right\rangle$$

Create all possible JUMP states produced from the operators

$$\downarrow$$

$$\left\langle \Phi_f \right|$$

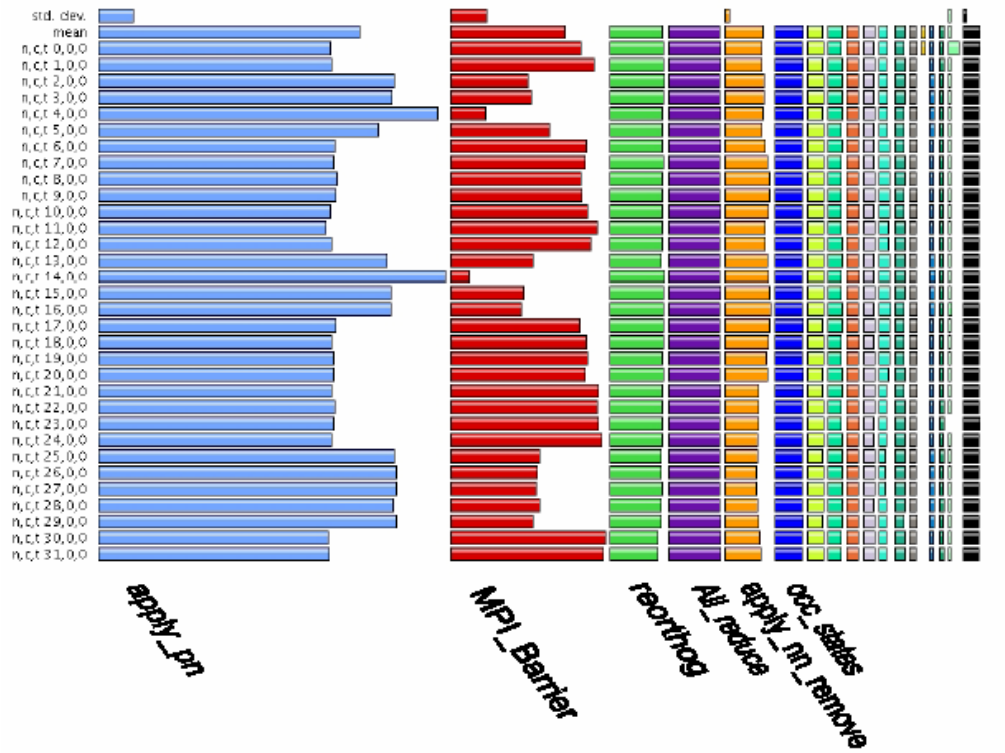
Search the final states for the jump state

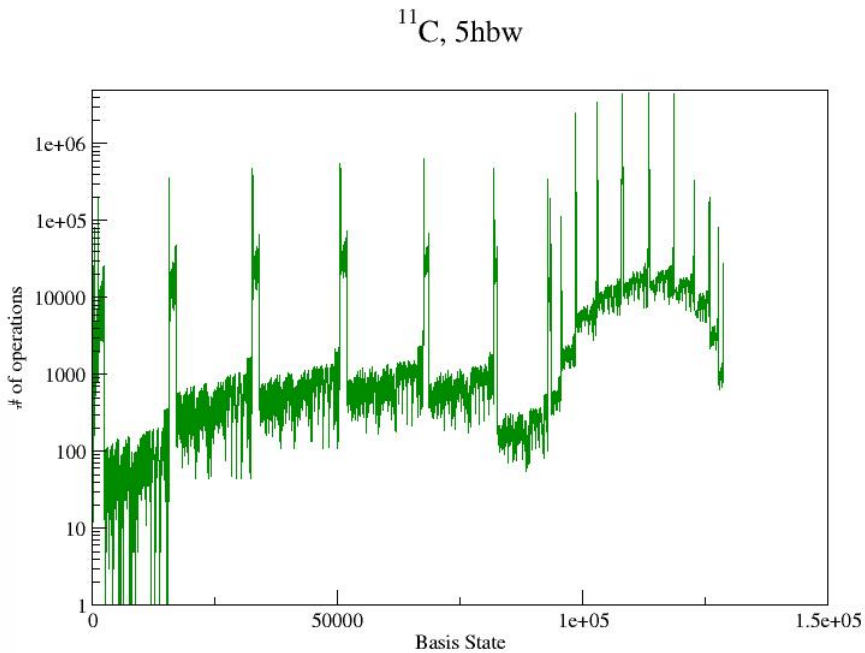
$N^2 \rightarrow N\log(N)$: Increase subroutine performance by a factor of 5-10 (depending on dim.)

Parallelization Analysis: Load Balancing

WORK DISTRIBUTION ANALYSIS

TAU Output



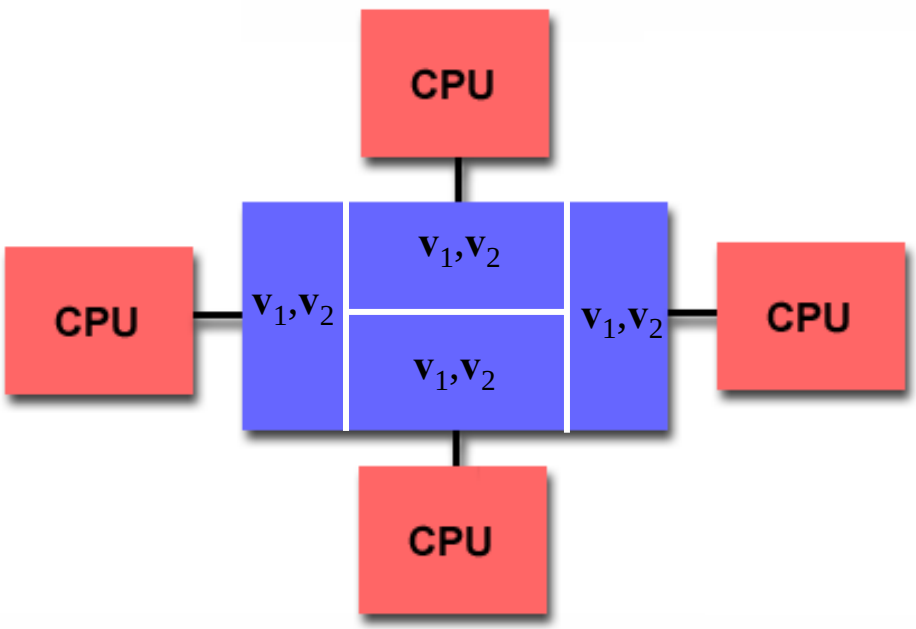


Parallelization

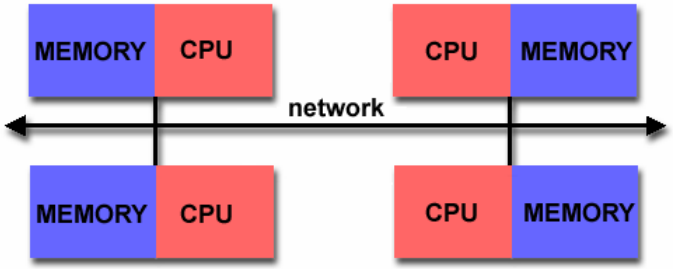
- **MPI 101**

$$H\mathbf{v}_1 = \mathbf{v}_2 \Rightarrow \begin{bmatrix} \text{Proc 1} \\ \text{Proc 2} \\ \vdots \\ \vdots \\ \vdots \\ \text{Proc n} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \\ \\ \\ \end{bmatrix}$$

Each CPU gets a copy of $\mathbf{v}_1$ and $\mathbf{v}_2$

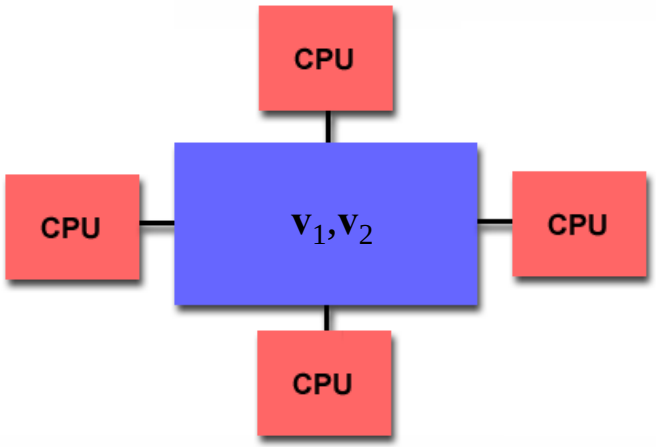


=



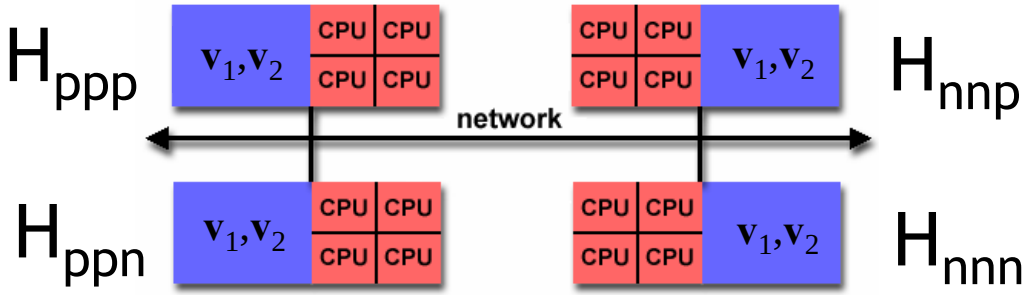
Efficient Parallelization

- **ARPACK (PARPACK) – optimized eigensolver (MFD)**
  - Limited by system resources
  - ^{10}B ~ 4 hours w/ 3500 processors



- **Hybrid Programming Model – MPI & OpenMP**
  - Decreases minimum memory requirement

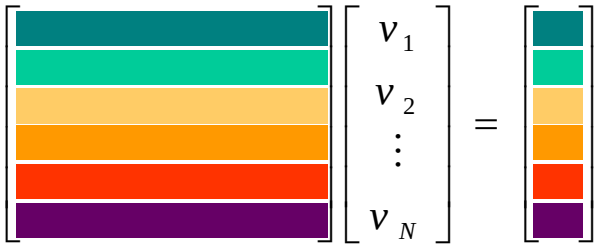
- **Single Instruction, Multiple Data → Multiple Instruction, Multiple Data**



What Else?

- **Reconsider algorithms for efficient parallelization**
  - Breaks up the large dimension vector

$$H\,\mathbf{v}_1 = \mathbf{v}_2 \Rightarrow$$



$\mathbf{v}_1$, $\mathbf{v}_2$ ~ 10^9, real(4)s

$\Rightarrow$ 8GB

Cannot be stored on 1 node

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} a_{11}v_1 + a_{12}v_2 + a_{13}v_3 \\ a_{21}v_1 + a_{22}v_2 + a_{23}v_3 \\ a_{31}v_1 + a_{32}v_2 + a_{33}v_3 \end{bmatrix} = \begin{bmatrix} | \\ \mathbf{a}_1 \\ | \end{bmatrix} v_1 + \begin{bmatrix} | \\ \mathbf{a}_2 \\ | \end{bmatrix} v_2 + \begin{bmatrix} | \\ \mathbf{a}_3 \\ | \end{bmatrix} v_3$$

  - Storage and on-the-fly
    - 1 proc starts with on-the-fly & another tries to retrieve m.e. If already calculated throw out else use retrieved values

The Frontier of NCSM Calculations

- **Two Body (m-scheme): 7 ≤ N ≤ 11, Nmax=10**
  N ≥ 12, Nmax = 6
  - Basis dimensions of 10^8 have been achieved
  - Nmax=8 requires restructuring of code
- **Three Body: Practical limit is Nmax = 6 for all p-shell nuclei**
  - Thus far results (Nmax=6 up to ^{6}Li (^{12}C?), Nmax=4 up to ^{13}C)
  - Our investigations include ^{15}O, ^{16}O, ^{17}O, ^{9}Be (4hbw… 6hbw?)
- **Four Body: Nmax = 4**
- **Not only limited by the number of matrix elements**
- **Other limiting factors**
  - Dimensions of the supporting vectors (1-body jumps, 2-body jumps, H_{ppp}, H_{nnn})
    - E.g ~ 4 billion two-body jumps (integer(4)) for ^{15}O, 6hbw = 16 GB
  - Lanczos vector storage

Future of HPC

- **Bigger and better?**
  - IBM Power6
  - If we need ~ 200,000 GB → (> 3000 nodes w/ 64 GB each)
- **Bring out the big guns!**
  - BlueGene/L
  - 65536 nodes / 131072 processors
  - 512 MB… yikes! → ~ 33,500 GB total
- **Hope**
  - DARPA, High Productivity Computing
  - Use what you've got wisely.
  - Collaborate with Computer Scientists

Summary

- **3-Body Forces are essential for ab initio nuclear structure calculations**
- **Computationally challenging due to memory and run-time performance limitations**
- **On-the-fly diagonalization methods provide greatest scaling potential (or a combination)**
- **Need efficient algorithms (suited for parallel optimization)**
- **Need to analyze workload distribution**
- **Need efficient parallelization schemes**
- **Considerable effort is still needed!!**

Acknowledgements

- **Calvin W. Johnson, SDSU**
- **W. Erich Ormand, N-Division, LLNL**
- **Greg Bronevetsky, CASC, LLNL**
- **Petr Navratil, N-Division, LLNL**
- **Department of Energy**