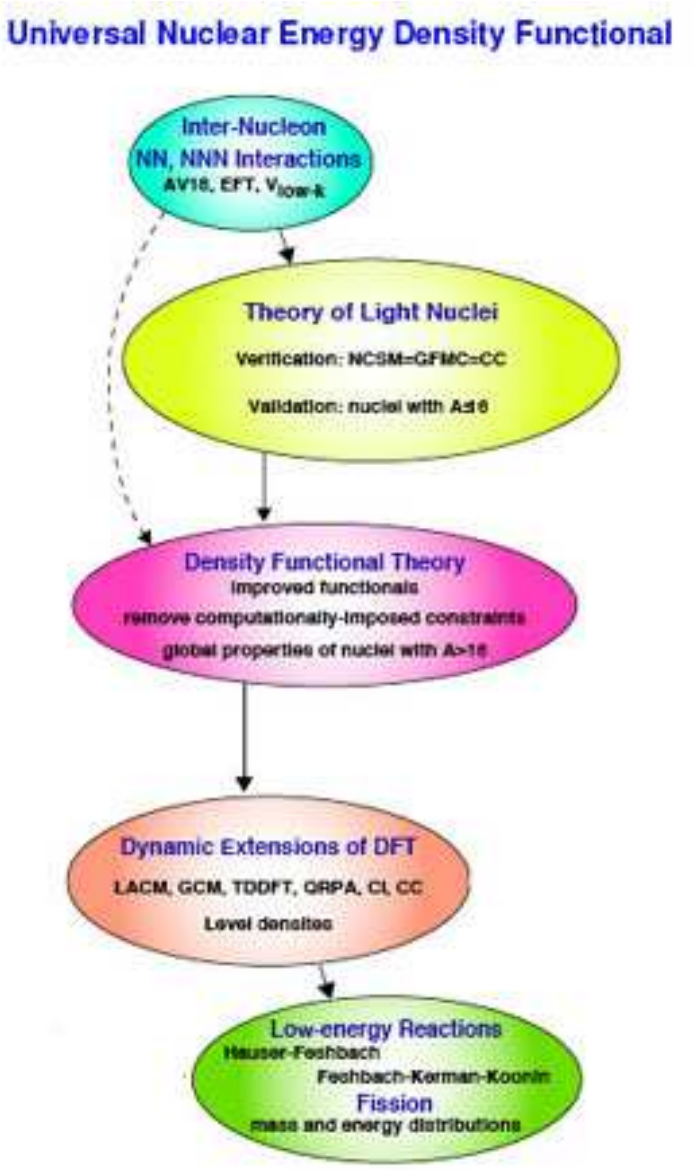
Many Fermion Dynamics – nuclear physics



IOWA STATE UNIVERSITY

Pieter Maris
pmaris@iastate.edu

Office of Science
INCITE
Advancing America's Science and Industrial Competitiveness

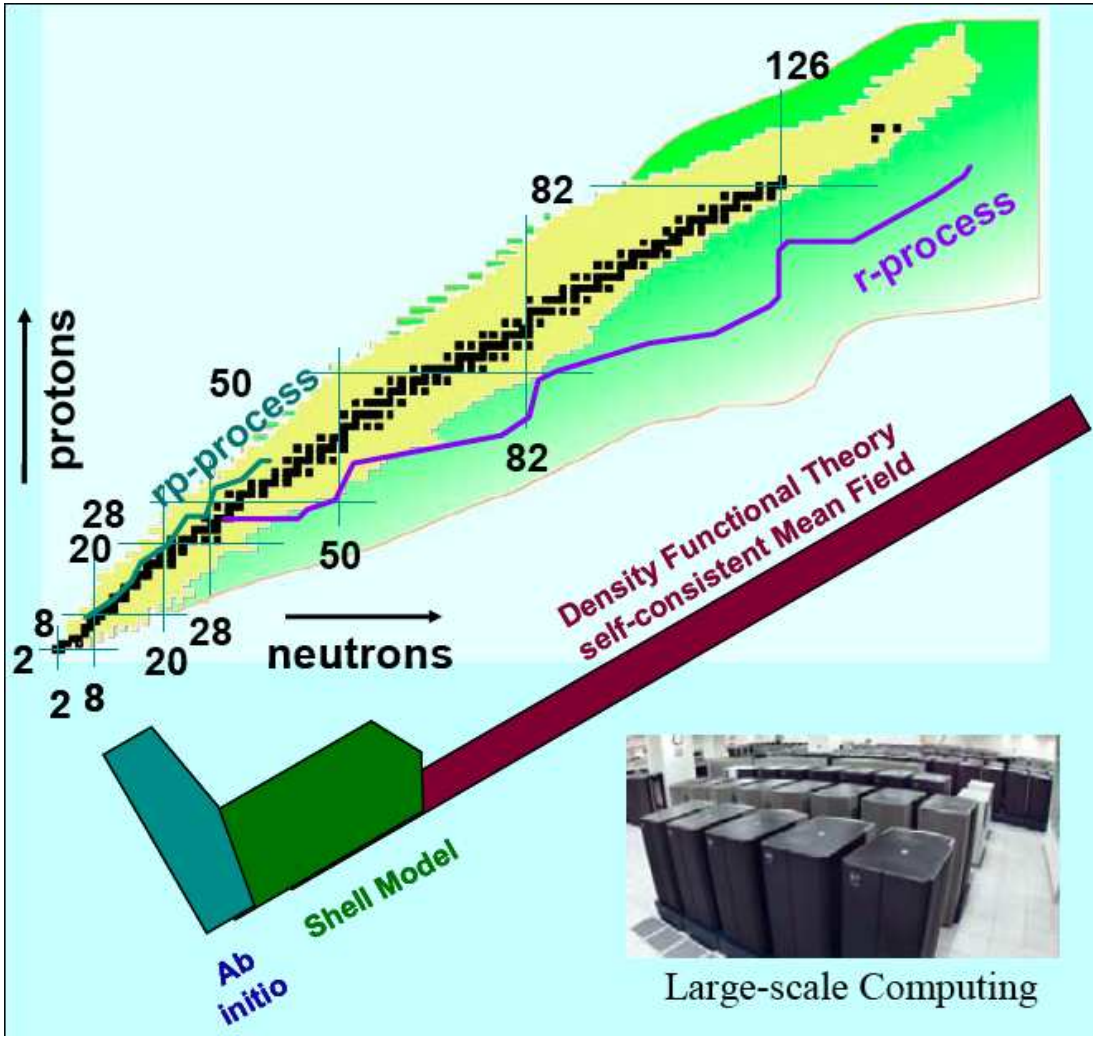
SciDAC
Scientific Discovery through Advanced Computing

Overview

- MFDn: Many Fermion Dynamics – nuclear physics
  - MFD: given a 2-body or 3-body interaction (potential) calculate mass spectrum and wave-functions of bound states of A fermions
    - nuclear physics: Nucleus A w. Z protons and N neutrons
  - goal: *ab initio* calculations up to $A \sim 50$ region
  - developed over last 25 years by James Vary contributions from several nuclear physicists no contributions from computer scientists until recently
  - part of SciDAC - UNEDF project (PI George Bertsch)
  - part of INCITE award on Jaguar (PI David Dean)
- Current MFDn collaborators
  - James Vary, Alina Negoita, PM (ISU)
  - Masha Sosonkina, Anurag Sharda (Ames Lab)
  - Esmond Ng, Chao Yang, Philip Sternberg (LBNL)
  - Petr Navratil (LLNL), Andrey Shirokov (Moscow)

Science

Uniform description of nuclear structure



- Nuclear Energy Density Functional based on *ab initio* calculations for light nuclei

- *Ab initio* calculations
  - Coupled Cluster (David Dean *et al*, ORNL)
  - MFDn (aka No Core Shell Model)
  - GFMC (Steve Pieper *et al*, ANL)

Computational Methods

- Expand nuclear wave function in H.O. basis functions
- Generate Many–Body basis
  - Slater-Determinant of single particle states
  - Carbon-12 using 8 H.O. levels: 33 million basis states
- Construct Many–Body Hamiltonian
  from 2-body (and 3-body) interactions
  - large symmetric real sparse matrix
  - bit-manupilation to determine nonzero m.e.
- Diagonalize Hamiltonian: solve for lowest 10 to 20 eigenvalues
  - iterative Lanczos algorithm w. orthogonalization
  - other diagonalization ideas: suggestions welcome
- Calculation of physical observables:
  - vector-vector and matrix-vector multiplications

Parallel Programming Model

- Fortran 90, MPI
- Until recently: completely self-contained, no use of libraries
  - future plan: explore use of math libraries
    - only use widely available libraries to ensure
      MFDn continues to run on a wide range of platforms
    - currently: implementing pARPACK for diagonalization
- Platform independent
  - NERSC: Seaborg (IBM-SP3), Bassi (IBM-SP5), Jacquard (Opteron)
  - ORNL: Jaguar (Cray XT3, XT4)
  - Livermore: Thunder (Itanium2), Atlas (Opteron)
  - Pittsburgh: BigBen (Cray XT3)
  - several university-based clusters (ISU, OSU, U. of Az)
  - future plan:
    - explore BlueGene?

IO Patterns and Strategies

- Input of 2-body and 3-body interaction files:
  - large files, same file read by all processors

    3-body interaction files for 8 H.O. levels: 2.5 Gb binary, 9 Gb formatted
- IO during run
  - relatively small files, unique to each processor
  - storage of matrix elements on local disk

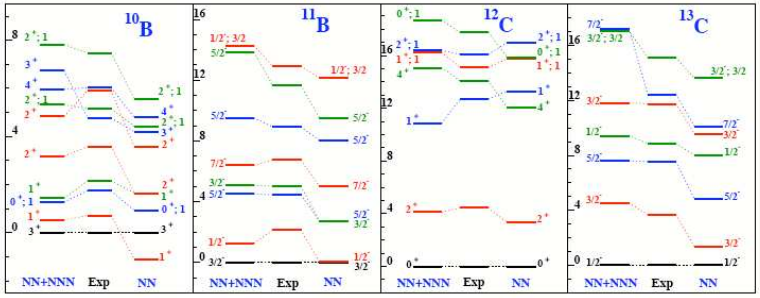
    allows for calculations that do not fit in memory
  - various restart options available
- Output near end of run: Nuclear wave functions
  - large files, written by processor 0 only

    Carbon-12 using 8 H.O. levels: 1 Gb binary, 6 Gb formatted
  - used as input for other programs: platform independent
  - current solution
    - written as binary (by processor 0 only)
    - follow-up run on reduced number of processors

      reads binary, writes platform independent wave functions

Visualization and analysis of output

- Mass spectrum – compared directly to experiment



- Primitive visualization (dates back to late nineties)
  of proton density, neutron density, matter density
  - should be updated and revisited
- Nuclear wave function input for other programs
  - TRDENS – Transition Densities
    - transitions between excited states and ground states
    - transitions between different nuclei

    (Petr Navratil, Livermore)
  - Nuclear Energy Density Functionals

    UNEDF SciDAC (PI: George Bertsch)

Performance and Debugging Tools

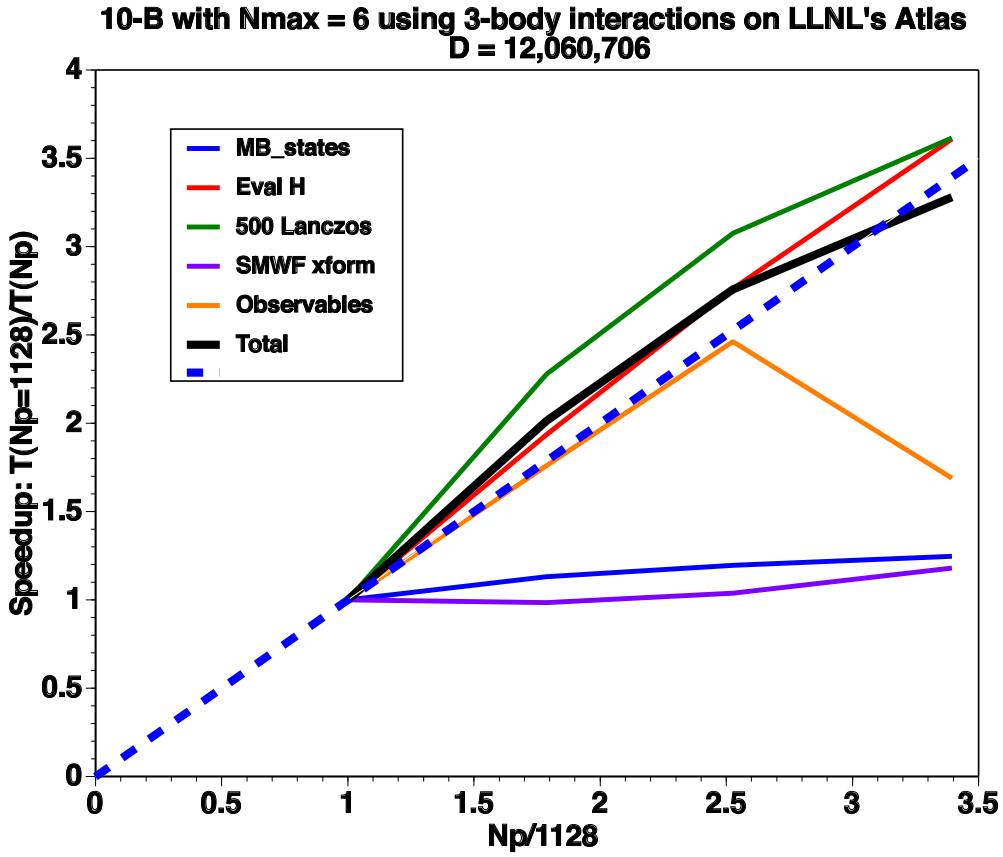
- Primitive use of performance and debugging tools
  - main debugging “tool”:
    extensive set of write statements, integrated in code,
    enabled by "verbosity" flags at compile time
  - main performance “tool”:
    timing of different sections of the code
  - limited use of IPM, Vampir, PAPI, Totalview

Performance and Debugging Tools

- Primitive use of performance and debugging tools
- Perceived performance bottlenecks
  - inner loop per processor performance
- Perceived scaling bottlenecks
  - algorithm for diagonalization
  - communication speed
- Potentially useful performance tool
  - reports on integer operations?
  - reports on bit-manipulations?
- As a user of a computing facility with limited resources
  interested in total (wallclock) time for given job
  not so much in percentage of peak performance

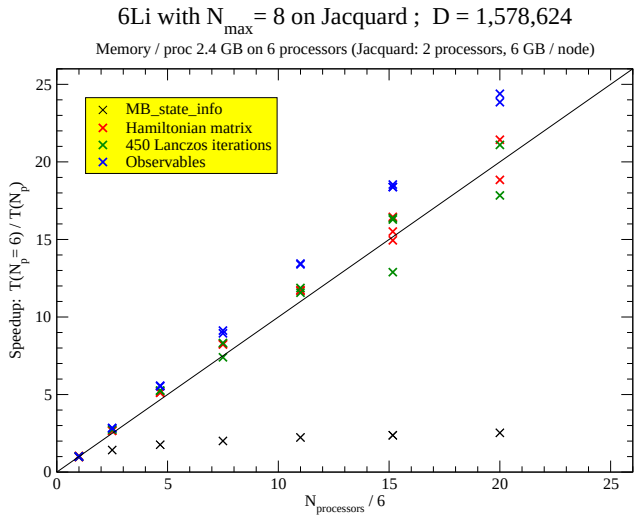
Scalability

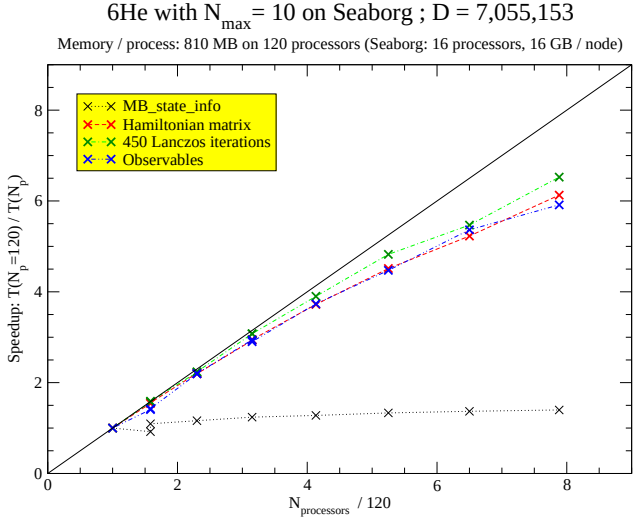
- Code has run okay up to 5,000 processors (Seaborg, Jaguar, Thunder, Atlas)
  - region of scaling starting at minimum number of processors for which problem fits in memory



Scalability

- Code has run okay up to 5,000 processors
- Number of processors restricted to $n(n+1)/2$
  - scaling achieved for n odd:
    optimal load balancing for major parts of code
  - code also runs on $n(n+1)/2$ with n even,
    but load balancing not ideal
- Eliminate non-scaling part of code
  - generate Many-Body basis on n procs (initial part of code)
  - run rest of program on $n(n+1)/2$ procs





Scalability

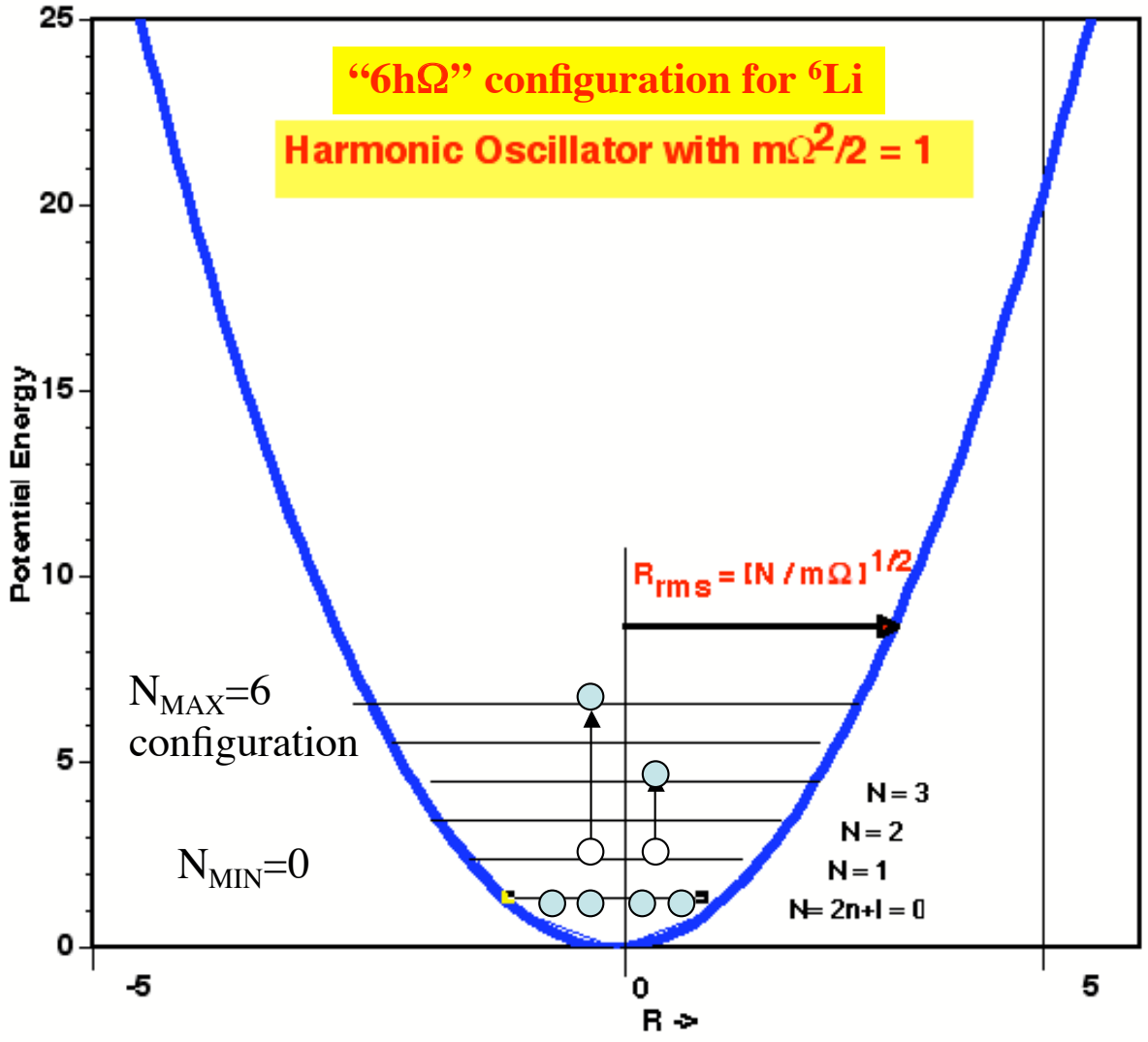
- Code has run okay up to 5,000 processors
- Goal in one year: running on 20k processors (Jaguar, Franklin)
- Top 5 pains
  1. Explosion of memory needed with increasing basis space
  2. Lack of code transparancy
     it being a legacy code dating back to early eighties
  3. Inner loop inefficiencies
  4. Size of 3-body interaction input files
  5. Output of wave functions (eigenvectors)
- Future plans:
  - to reduce memory requirements:
    - IO to local disk of matrix elements
    - recompute matrix elements “on the fly”
    - compression matrix (and vector) arrays
  - explore parallel IO (MPI_IO)

Roadmap

- Science goal for next 2 years
  - Carbon-12 and Oxygen-16 using 10 H.O. levels: D = 594,496,743 and D = 996,878,170
  - Establish convergence w.r.t. basis space truncation for light nuclei
  - Explore nuclei up through $A \sim 50$
- Improvements needed in code
  - per processor performance
  - memory management
- Plans
  - utilize parallel math libraries
  - address inner loop inefficiencies
  - parallel IO, IO to local disk
  - explore different diagonalization algorithms

MFD_nuclear – Basis space

- Harmonic oscillator basis (Shell Model)



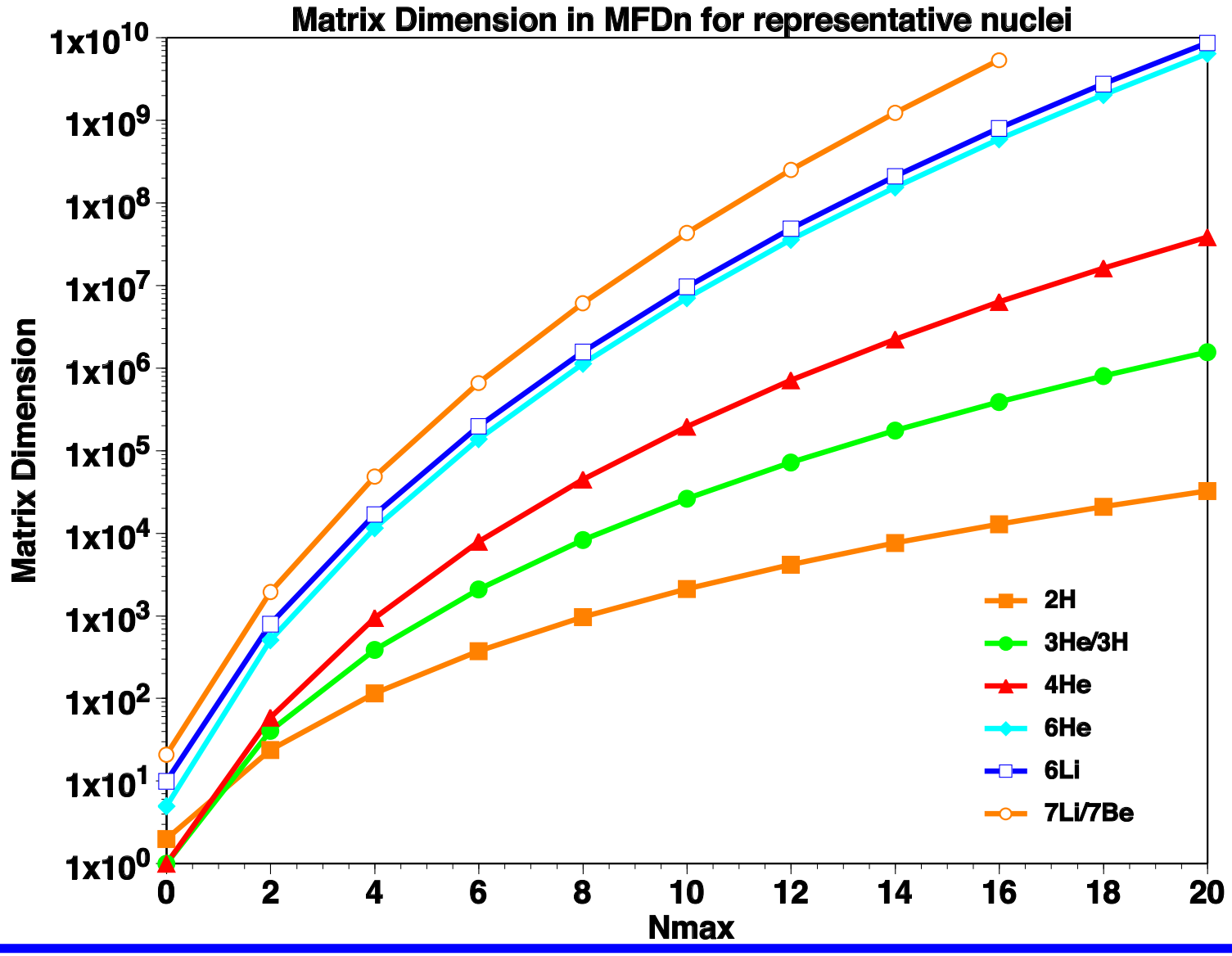
MFD_nuclear – Basis space

- Harmonic oscillator basis (Shell Model)
- Product space of single-particle H.O. states

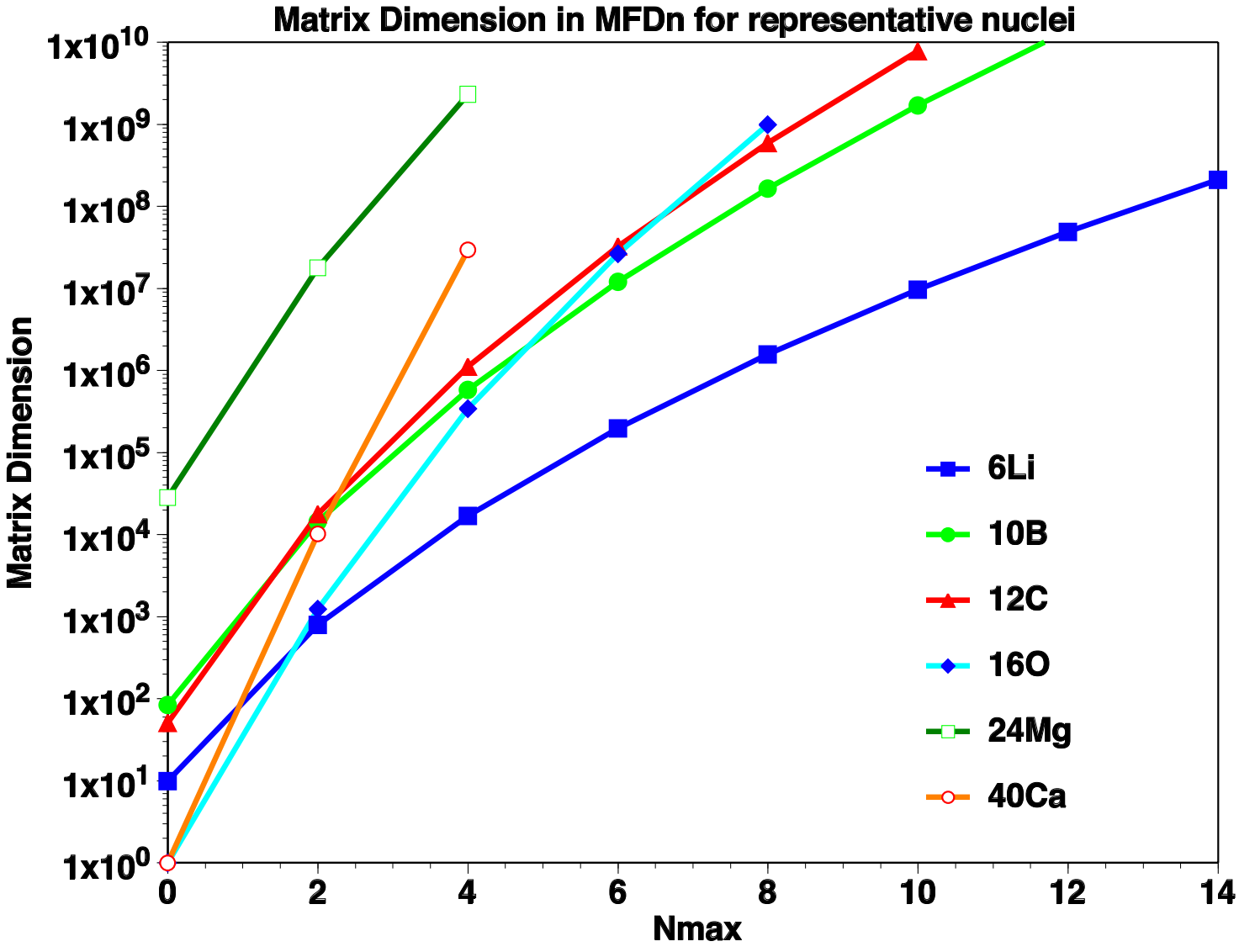
$$|\Psi\rangle \quad = \quad |\psi_1\rangle \otimes \ldots \otimes |\psi_n\rangle$$

- Uses single-particle coordinates
  not relative (Jacobi) coordinates
  - straightforward to extend to many particles
  - Pauli exclusion principle:
    Implementation of Fermi statistics for identical particles
    in systems with many fermions, $A > 4$
    much easier in single-particle coordinates
    than in relative coordinates

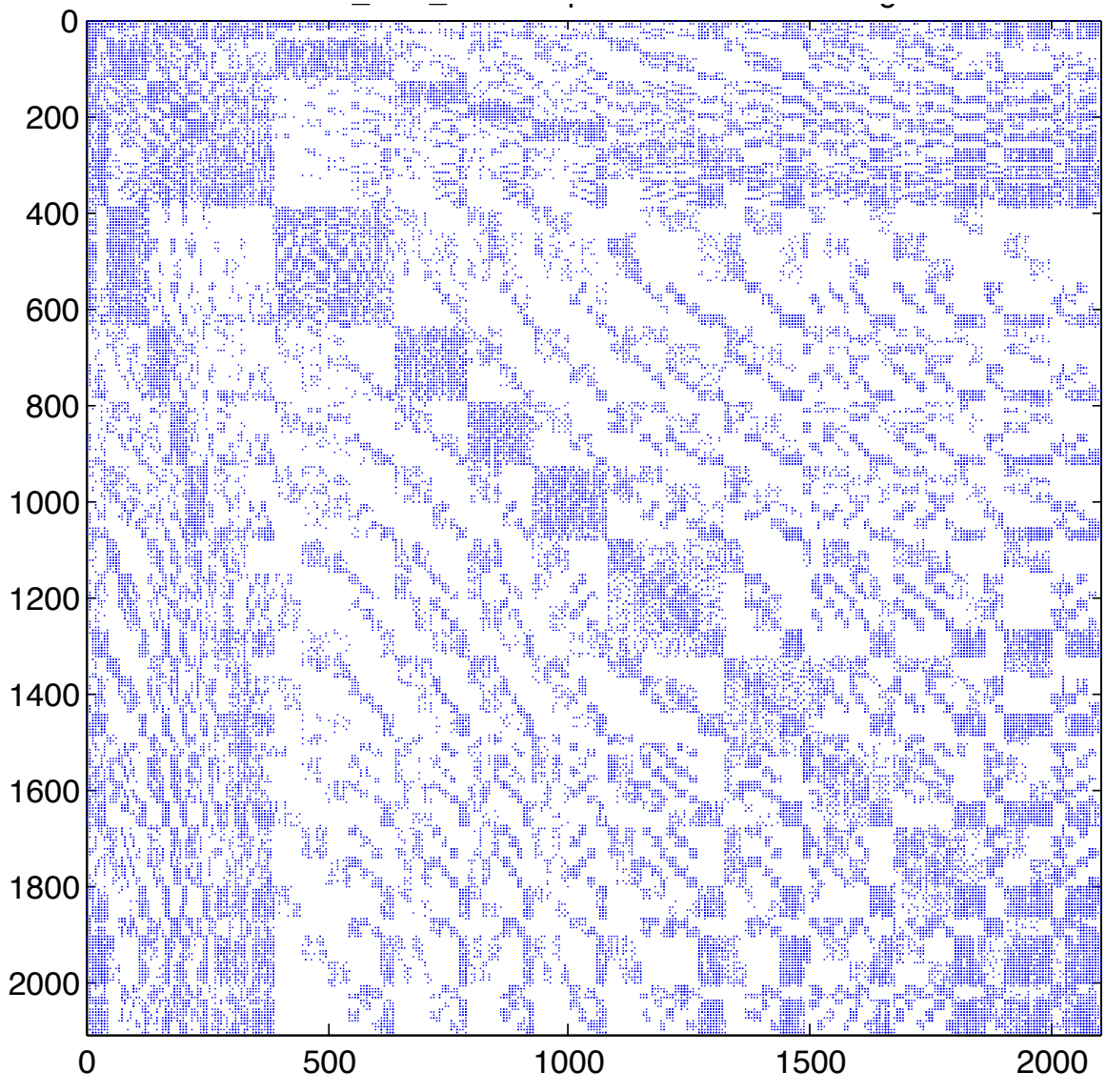
MFD_nuclear – matrix dimensionality



MFD_nuclear – matrix dimensionality



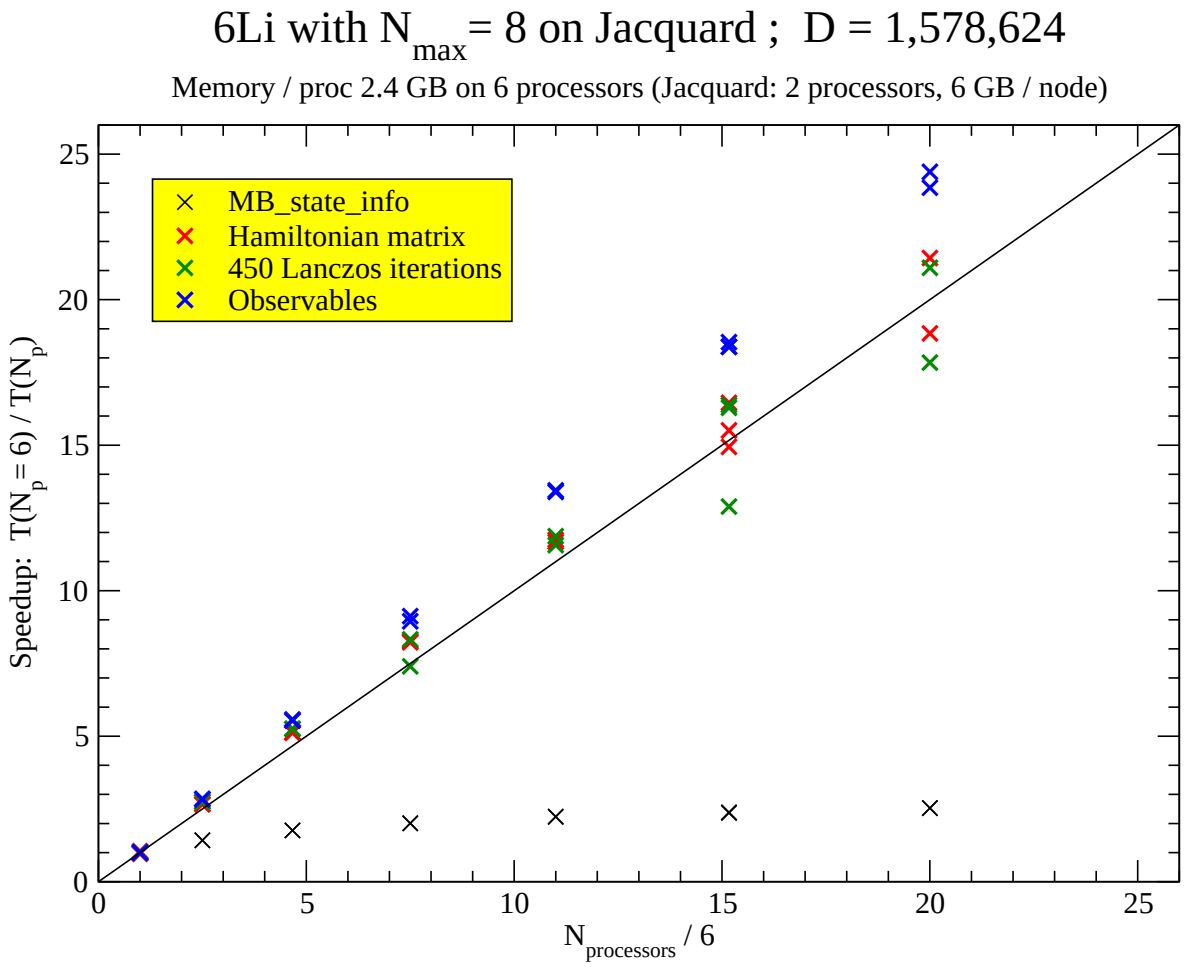
MFD_nuclear – matrix structure



Scalability

Straightforward to eliminate non-scaling part of code

- generate Many-Body basis on n procs (initial part of code)
- run rest of program on $n(n+1)/2$ procs

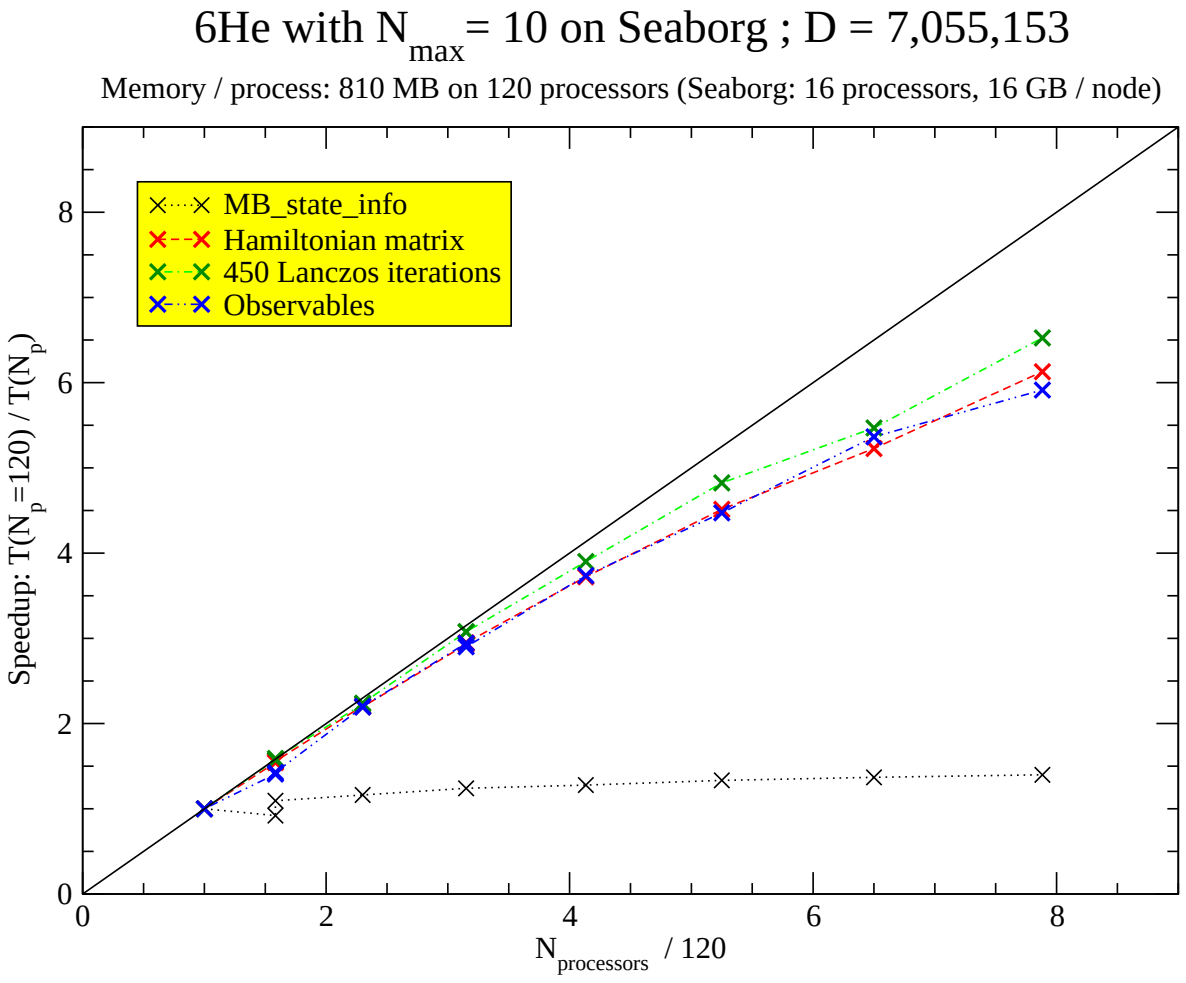


reference run
on 6 processors
poorly balanced;
load balancing
improves significantly
with number of procs

Scalability

Straightforward to eliminate non-scaling part of code

- generate Many-Body basis on n procs (initial part of code)
- run rest of program on $n(n+1)/2$ procs

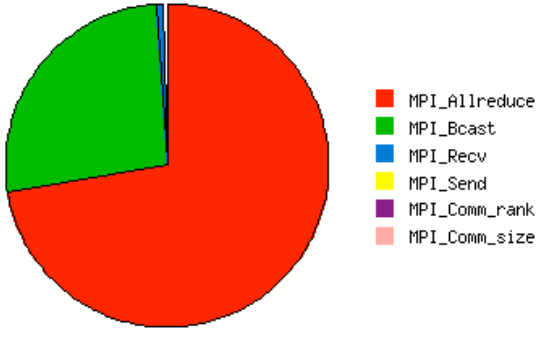


MFD_nuclear – IPM

MPI time statistics - 28 tasks

Call	Sum over all tasks	Average (per task)	Task CV (%)	Task Minimum	Task Maxium	% of MPI	% of wall
MPI_Allreduce	1.074e+03	3.834e+01	107.64	1.298e+01	1.126e+02	72.306	8.642
MPI_Bcast	3.955e+02	1.413e+01	55.70	6.145e+00	2.794e+01	26.639	3.184
MPI_Recv	9.181e+00	3.279e-01	378.86	1.053e+00	1.638e+00	0.618	0.074
MPI_Send	6.492e+00	2.318e-01	67.89	7.708e-02	4.929e-01	0.437	0.052
MPI_Comm_rank	1.547e-04	5.526e-06	23.22	3.815e-06	9.060e-06	0.000	0.000
MPI_Comm_size	8.130e-05	2.904e-06	40.91	1.907e-06	6.437e-06	0.000	0.000

Percent of total MPI Time



MPI call frequency statistics - 28 tasks

Call	Tasks	Number of Calls (per task)					Buffer Size (Bytes)		
		Sum	Average	CV (%)	Minimum	Maxium	Sum	Avg. per task	Avg. per call
MPI_Allreduce	28	3.813e+05	1.362e+04	1.80e+00	1.348e+04	1.410e+04	0.000e+00	0.000e+00	0.000e+00
MPI_Bcast	28	7.708e+04	2.753e+03	0.00e+00	2.753e+03	2.753e+03	0.000e+00	0.000e+00	0.000e+00
MPI_Recv	7	1.761e+03	6.289e+01	3.77e+02	2.400e+02	3.210e+02	0.000e+00	0.000e+00	0.000e+00
MPI_Send	28	1.761e+03	6.289e+01	3.99e+01	1.600e+01	9.000e+01	0.000e+00	0.000e+00	0.000e+00
MPI_Comm_rank	28	2.800e+01	1.000e+00	0.00e+00	1.000e+00	1.000e+00	0.000e+00	0.000e+00	0.000e+00
MPI_Comm_size	28	2.800e+01	1.000e+00	0.00e+00	1.000e+00	1.000e+00	0.000e+00	0.000e+00	0.000e+00

MFD_nuclear – Vampir