SLDA and TD-SLDA on JaguarPF

K. J. Roche
High Performance Computing Group
Pacific Northwest National Laboratory
Nuclear Theory Group
University of Washington
In collaboration with :
Aurel Bulgac
Michael Forbes
Alan Luo
Piotr Magierski
Ionel Stetcu
Sukjin Yoon
Yongle Yu

U.S. DEPARTMENT OF ENERGY
Office of Science

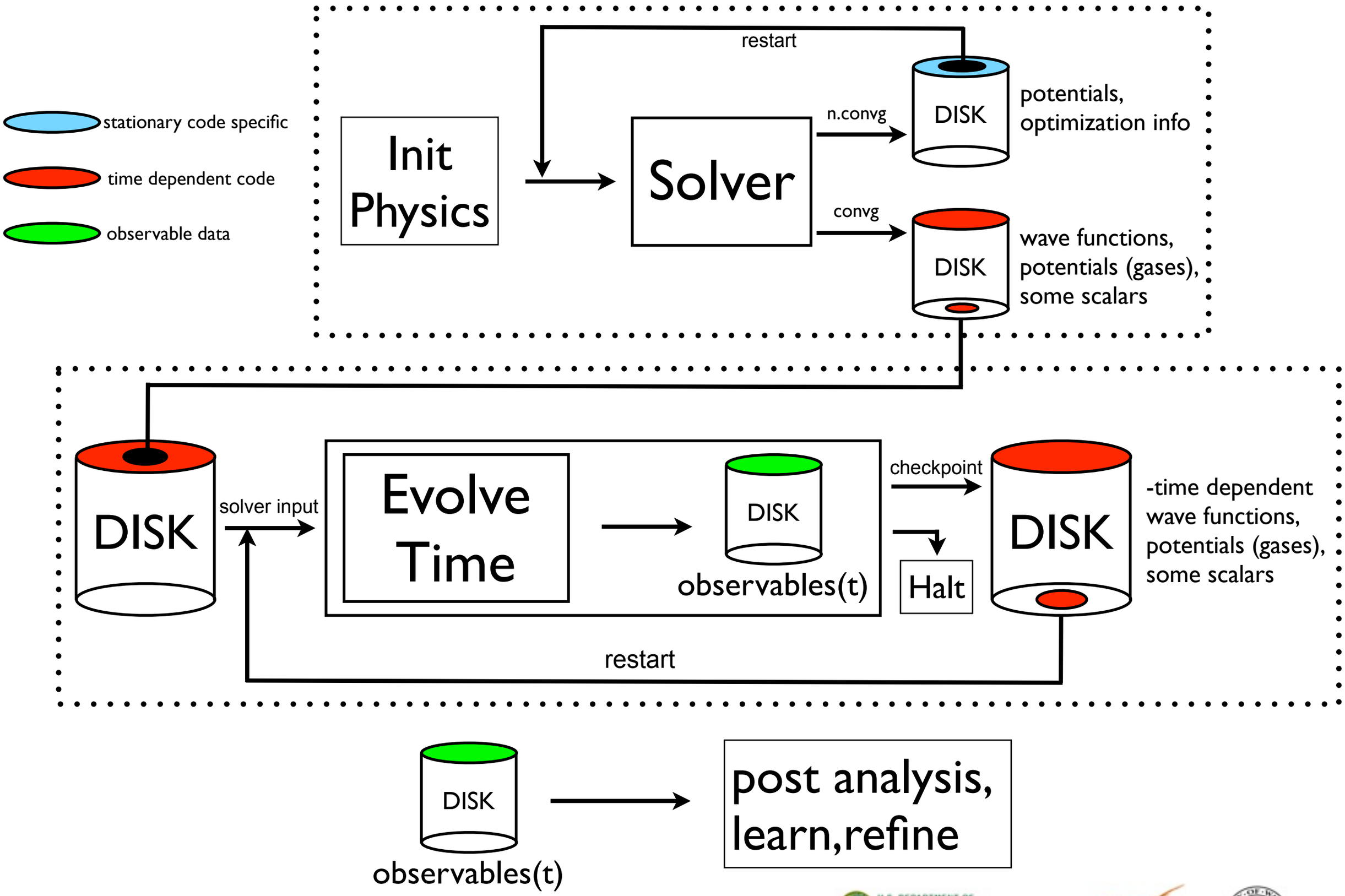
Pacific Northwest NATIONAL LABORATORY

JaguarPF and Trends in Target Computing Platforms

Hex-Core AMD Opteron (TM)	2.6e9 Hz clock	4 FP_OPs / cycle / core 128 bit registers
PEs	18,688 nodes	224,256 cpu-cores (processors)
Memory	16 GB / node 6 MB shared L3 / chip 512 KB L2 / core 64 KB D,I L1 / core	dual socket nodes 800 MHz DDR2 DIMM 25.6 GBps / node memory bw
Network	AMD HT SeaStar2+	3D torus topology 6 switch ports / SeaStar2+ chip 9.6 GBps interconnect bw / port 3.2GBps injection bw
Operating Systems	Cray Linux Environment (CLE) (xt-os2.2.41A)	SuSE Linux on service / io nodes

FY	Aggregated Cycles	Aggregated Memory	Aggregated FLOPs	Memory/FLOPs
2008	65.7888 THz	61.1875 TB	263.155 TF	0.2556
2009	343.8592 THz	321.057 TB	1.375 PF	0.2567
2010	583.0656 THz	321.057 TB	2.332 PF	0.1513

Structure of the SLDA Production Software



A Small-scale DEMO of the Tool : Unitary Fermi Gas in Trap + Ball and Rod Stirring

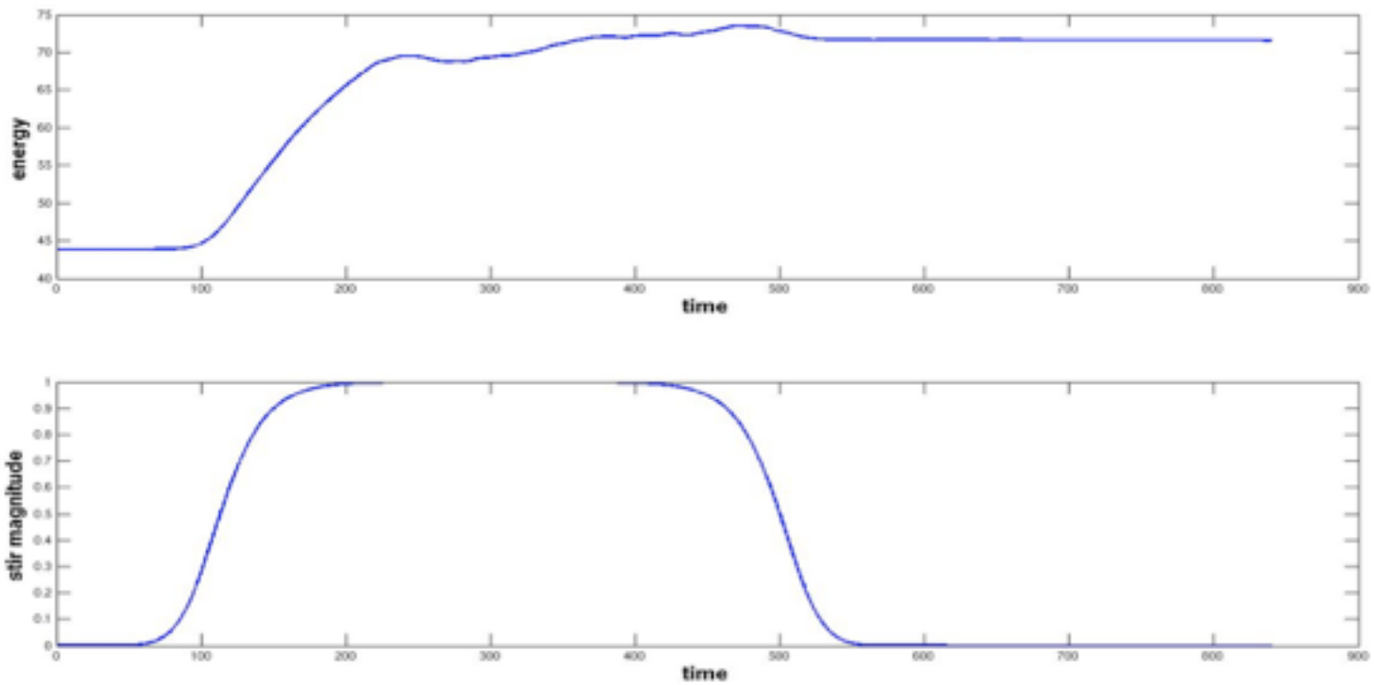
300 particles ; 32^3 lattice ; 104,132 time steps ; 1509 I/O events of analysis data

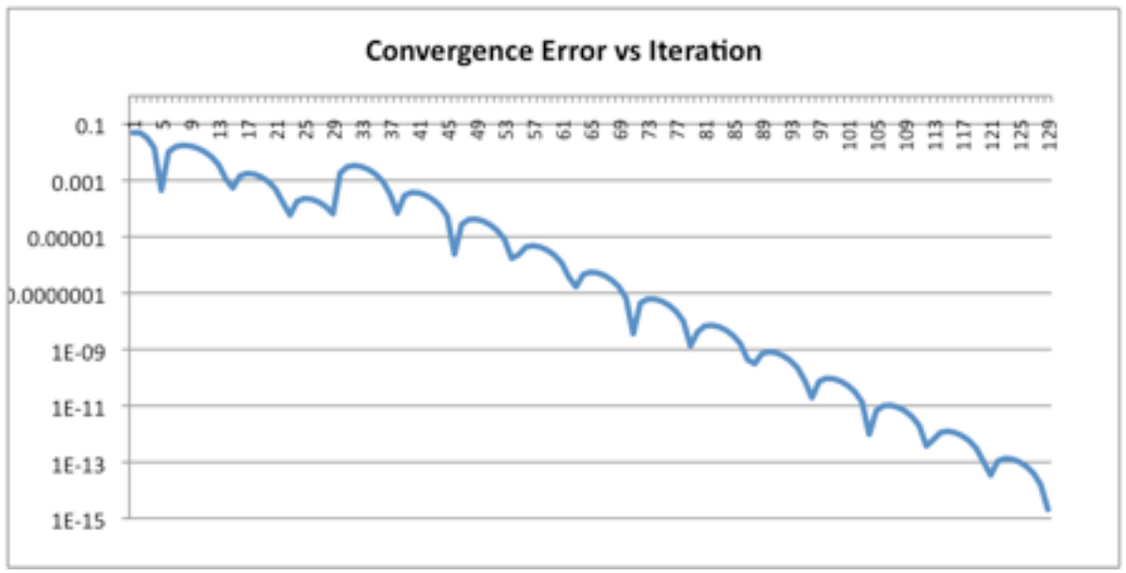
Run 1 self-consistent solver generated stationary solutions for the system

Run 2 initialized the TD code, executed a total **87,271** time steps, **1264** completed I/O events, check pointed

Run 3 restarted at time step **87,272**, executed **16,861** additional time steps, **245** additional I/O events, exited cleanly

	Run 1	**Run 2**	**Run 3**	Total
PEs	612	**9458**	**9458**	
Time(s)	451.081	**34,909.812**	**11,020.127**	46,381.02
CPU $(hours)	76.68377	**91,715.83386**	**28,952.32255**	120,744.8401

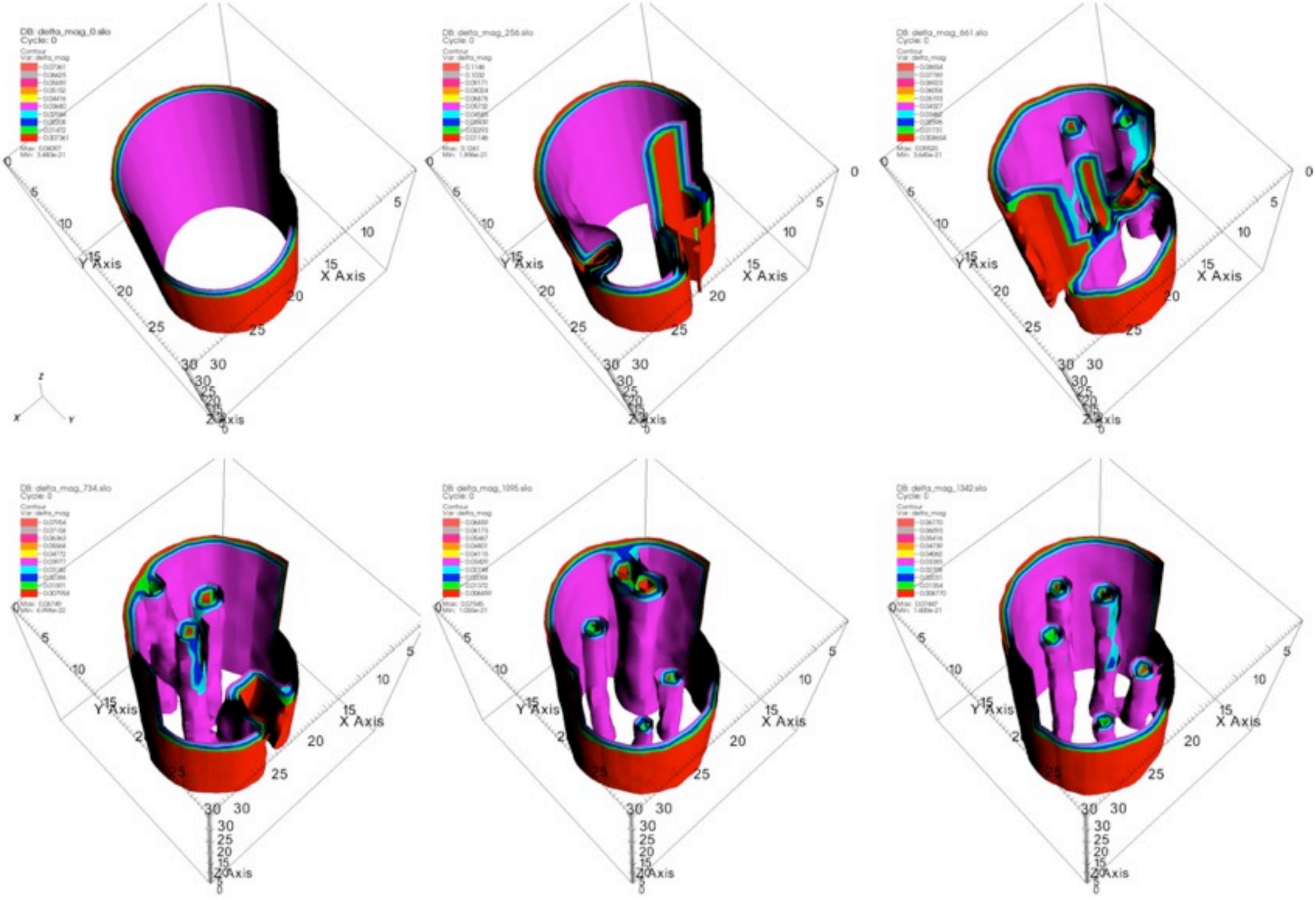




U.S. DEPARTMENT OF ENERGY
Office of Science

Pacific Northwest NATIONAL LABORATORY

Numerical Studies of Vortex Formation and Dynamics in Superfluid Fermi Systems



Benchmark(1) on JaguarPF for ASCR's OMB PART Software Effectiveness Metric: UG Problem

5216 particles ; 103,917 wavefunctions ; 50x50x100 lattice ; 2,051 (100K) time steps ; 26 I/O events of analysis data

Solver (Run1)
- 51 parallel , parallel groups
- 144 PEs / group
- simultaneous vs in sequence, perfect strong scaling
- 129 iterations to converge

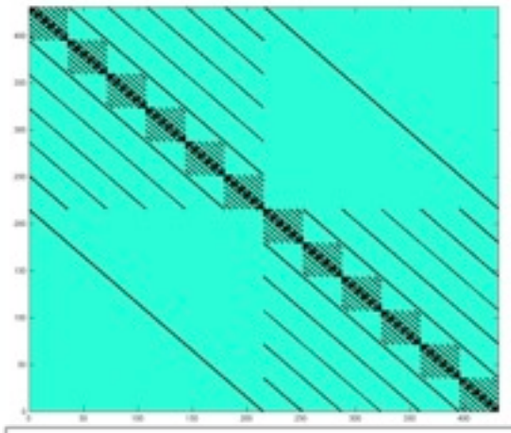
TD (Run2)
- initialize (read soln) TD code
- 1 time step
- 1 data analysis io event
- checkpoint TD wavefunctions
  - 8TB data written
  - 24 Lustre write groups

TD (Run3)
- restart TD code
- 2050 time steps
- 25 data analysis io events
- clean exit

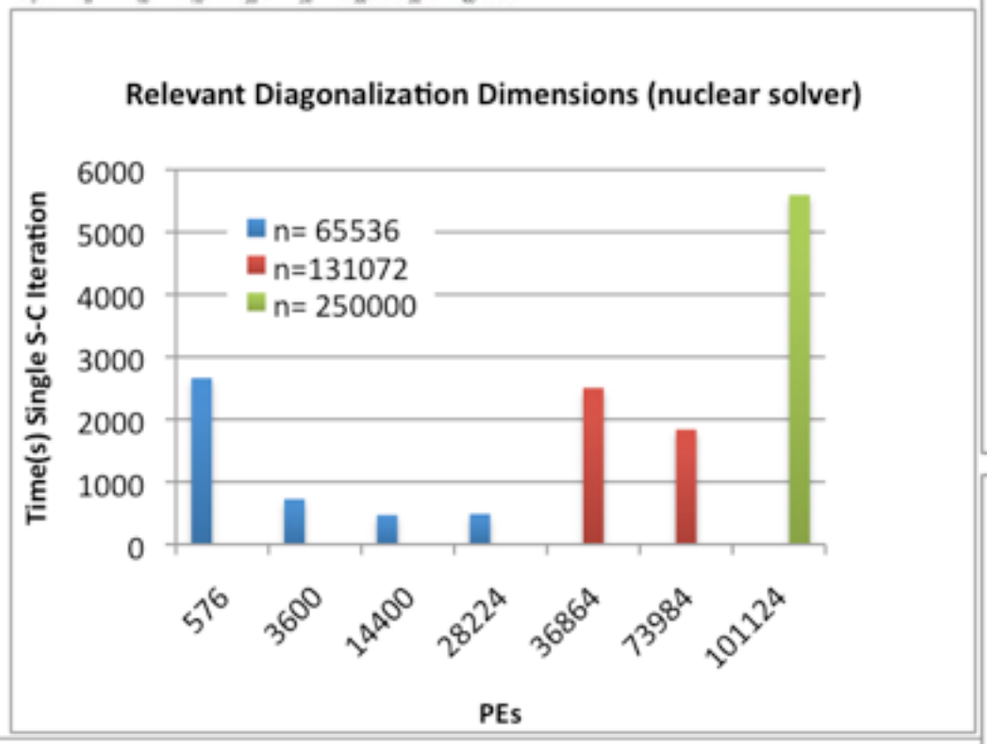
Machine Data	Run 1	**Run 2**	**Run 3**	Total
Instructions	3.335083409e17	**3.538162667e18**	**4.415760819e18**	8.287431827e18
Floating Point Ops	3.628450357e15	**1.91882289e14**	**3.235240267e17**	3.273443593e17
Wall Time(s)	11,085.29975	**8,291.248311**	**12,913.89644**	32,290.4445
CPU $(hours)	22,614.01149	**239,333.7919**	**372,770.3823**	634,718.1857
PEs	7,344	**103,917**	**103,917**	

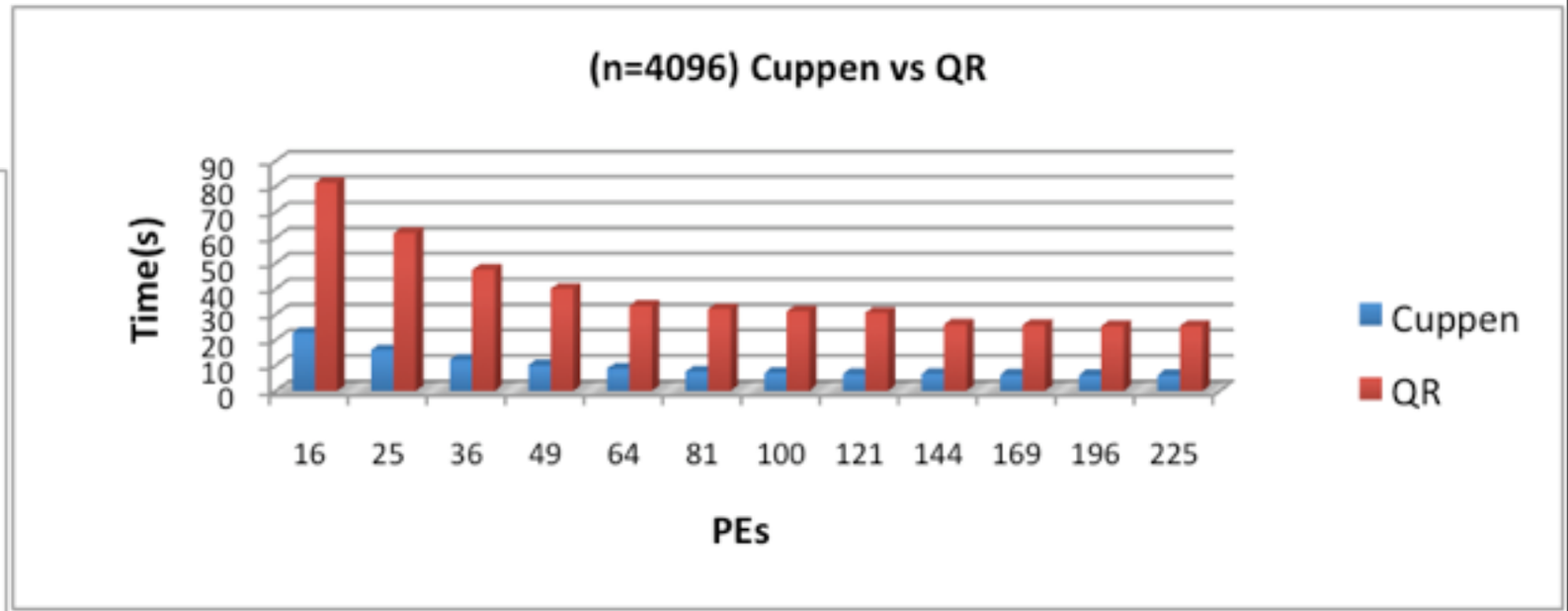
In Q3, successfully benchmarked the TD code on 217,752 2-component qpwf system on 62^3 lattice. - over 97% of the complete system!

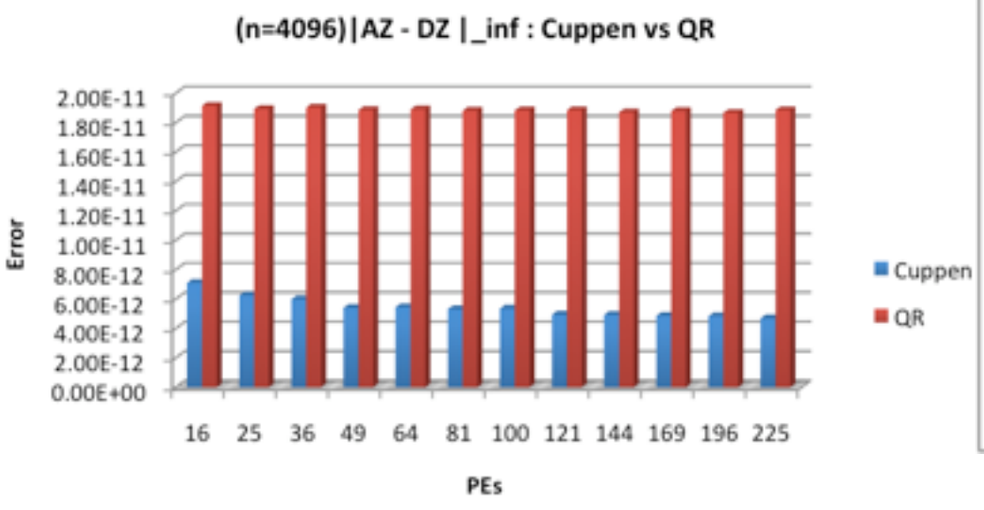
Users Beware ... Some Learned Lessons

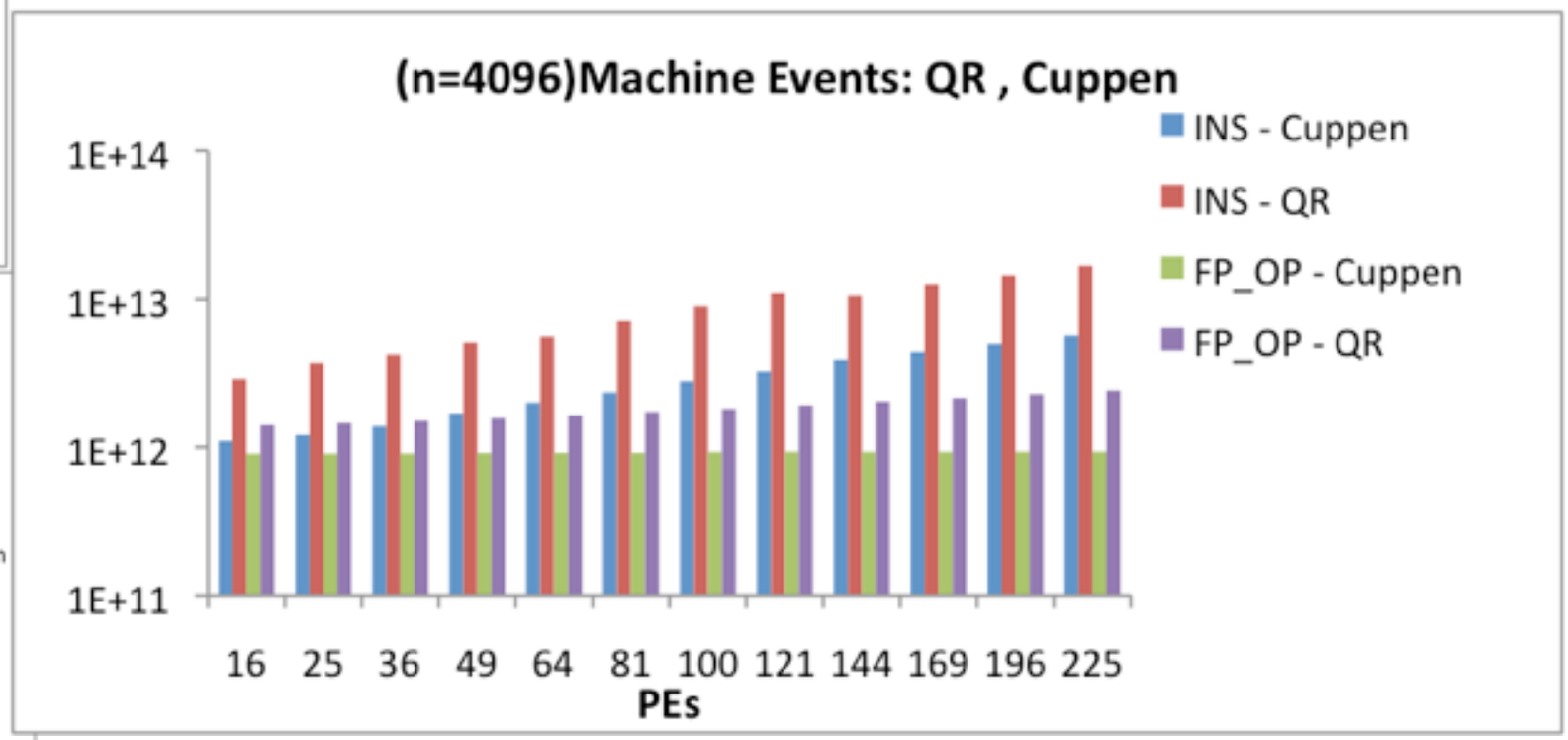


- J.J.M. Cuppen, *A Divide and Conquer Method for the Symmetric Tridiagonal Eigenproblem*, Numer. Math. 36, 177-195 (1981)
- F. Tisseur and J.J. Dongarra, *Parallelizing the Divide and Conquer Algorithm for the Symmetric Tridiagonal Eigenvalue Problem on Distributed Memory Architectures*, lawn132 (1998)









U.S. DEPARTMENT OF ENERGY
Office of Science

Pacific Northwest NATIONAL LABORATORY

Aside on FILEs and IO

ANSI C
- stream of BYTEs
- points to a FILE structure
- fopen,fwrite,fread,fclose

```
void f_copn_ ( char * ffn , int * ffd , int * len ) ;

void f_ccls_ ( int * ffd ) ;

void f_crm_ ( char * ffn , int * len ) ;

void f_cwr_ ( int * ffd , void * fbf , int * fsz , int * nobj , int * ierr ) ;

void f_crd_ ( int * ffd , void * fbf , int * fsz , int * nobj , int * ierr ) ;
```

```
typedef struct  {
        int             level;      /* fill/empty level of buffer */
        unsigned        flags;      /* File status flags          */
        char            fd;         /* File descriptor            */
        unsigned char   hold;       /* Ungetc char if no buffer   */
        int             bsize;      /* Buffer size                */
        unsigned char   *buffer;    /* Data transfer buffer       */
        unsigned char   *curp;      /* Current active pointer     */
        unsigned        istemp;     /* Temporary file indicator   */
        short           token;      /* Used for validity checking */
}       FILE;
```

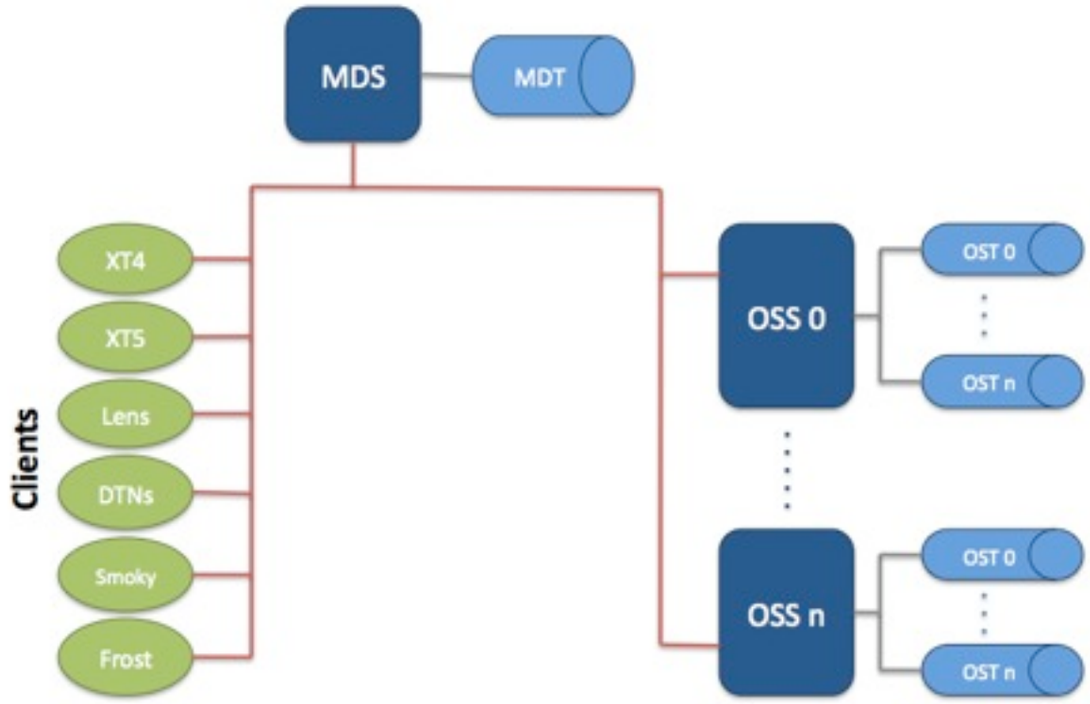
Fortran
- sequence of records
- open,write,read,close
- IOLENGTH , RECL

```
fn = '/tmp/work/roche/mpt-omp/ben.txt'//
CHAR(0)

call f_copn ( fn , fd , LEN( fn ) )

call f_cwr ( fd , a , 16 , ndim , ierr )

call f_ccls ( fd )

call f_copn ( fn , fd , LEN( fn ) )

call f_crd ( fd , a_bk , 16 , ndim , ierr )

call f_ccls ( fd )

call f_crm ( fn , LEN( fn ) )
```

```
0 -rw-r--r-- 1 roche roche  1608 2010-06-21 21:03 fortran-dat.bn
0 -rw------- 1 roche roche  1600 2010-06-21 21:03 c-data.dat
```

Aside on FILEs and IO (2)

POSIX (UNIX)
- stream of BYTES
- file descriptors
  - -index into file descriptor table
  - -kept in user process
    - -points to entry in system in-memory inode table
- open,write,read,close, ioctl



Spider (Lustre) :

- MDS, file names and directories in the filesystem, file open, close, state mgt
- OSS, provides file service, and network request handling for set of OSTs
- OST, stores chunks of files as data objects -may be stripped across one or more OSTs
  - -Spider has 672 OSTs
  - -7 TB per OST
  - -1 MB Default stripe size
  - -4 Default OST count

Aside on FILEs and IO (3) -SLDA Approach

```
module load liblut ; -LUT

lut__open() ;

lut__close() ;

lut_putl() ;

pwrite() ;


pread() ;
```

- form modulo classes from MPI communicator over the number of I/O groups
  - for both proton and neutron communicators in nuclear case (44 for protons, 44 for neutrons)

- fit the stripe size to the largest single data item if possible
  - eg for nuclear code and 32^3 lattice, a single 4-component term is 4 * 32^3 * 16 / 2^20 = 2MB

- set the stripe pattern (I use round-robin) and number of target OSTs (I use 88 in nuc code) for target PATH / FILE
  - eg lfs setstripe /tmp/work/roche/kio -s 2m -i -1 -c 88

Dominant I/O Demands for Checkpoint / Restart of Software:

unitary : 22 * NWF * Nx * Ny * Nz * sizeof(double complex)

-eg 22 * 103917 * 50 * 50 * 100 * 16 ~ 8,517 GB or ~8.317 TB

nuclear : 44 * NWF * Nx * Ny * Nz * sizeof(double complex)

-eg 44 * 43366 * 32^3 * 16 ~ 931.691 GB

Performance: POSIX ~ [225,350]MBps , my use of LUSTRE ~ [5,10]GBps

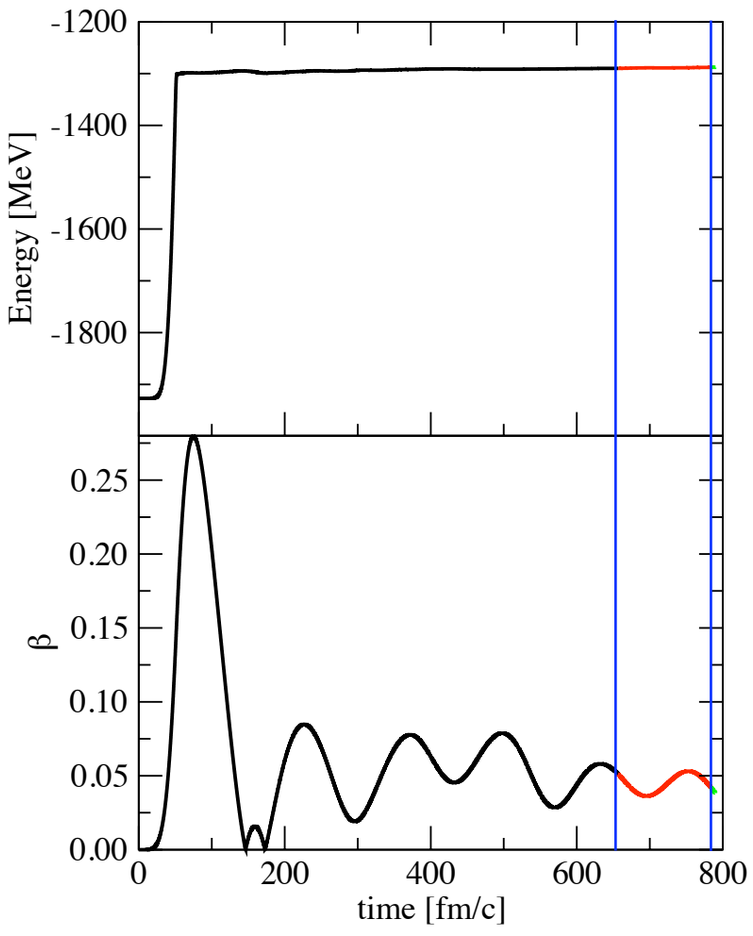
Another DEMO of the Tool : Nuclear LACM

280Cf ; 32^3 lattice ; 43,380 4-Component QPWFs, 13,173 time steps ; 1317 I/O events of analysis data

Run 1 initialized the TD code, executed a total **10,951** time steps, **1095** completed I/O events, check pointed

Run 2 restarted at time step **10,952**, executed **2,121** additional time steps, **212** additional I/O events, check pointed

Run 3 restarted at time step **13,073**, executed **101** additional time steps, **10** additional I/O events, exited cleanly



Run	Wall(s)
1	22,223
2	3,746
3	302

← Total wall times

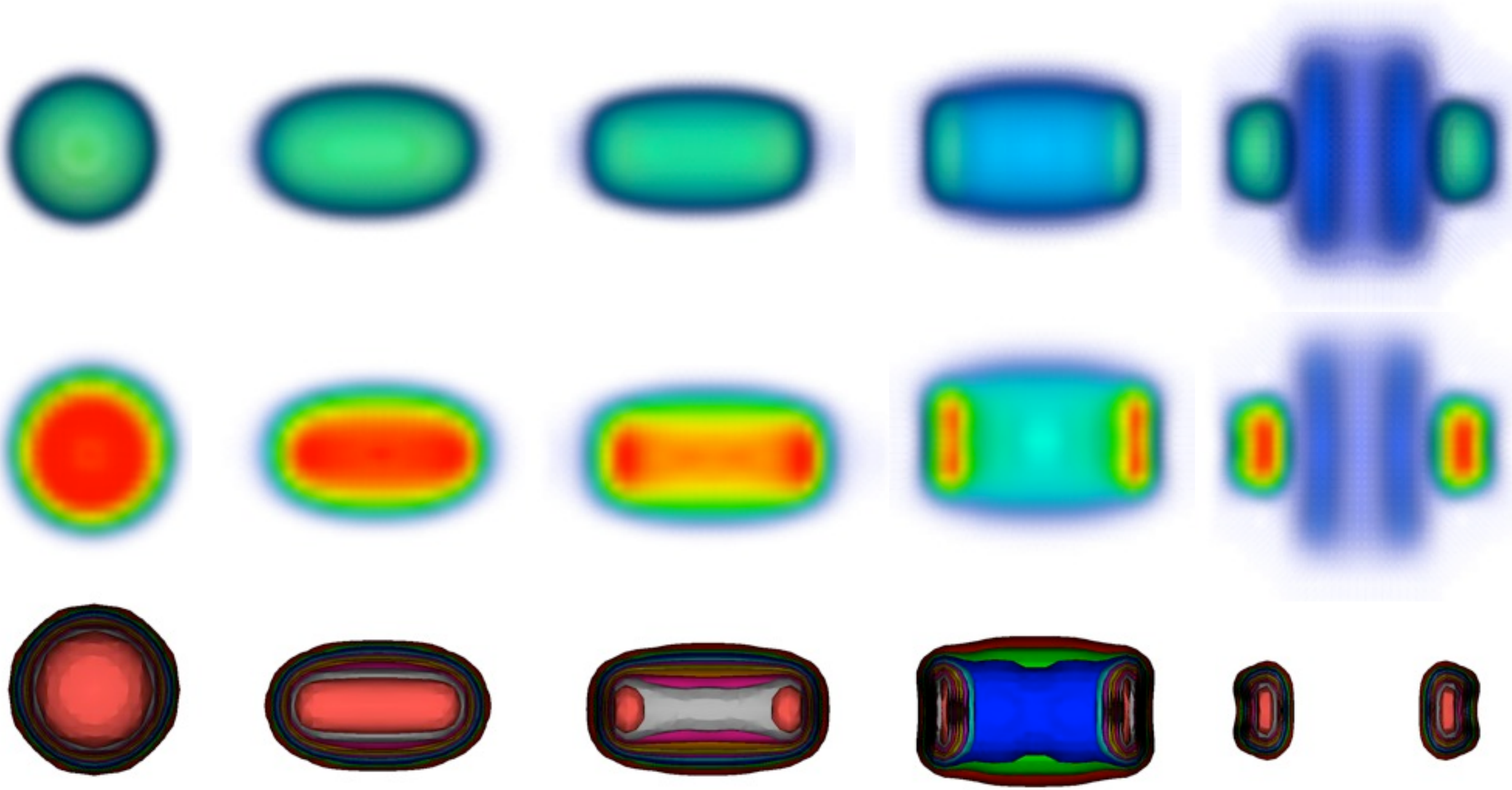
↓ Time loop times

Run	Time(s)	INS	FP_OPs	L2DCM
1	20367	2.0028E+18	8.7006E+16	2.2722E+15
2	3528	3.1354E+17	1.6852E+16	4.3962E+14
3	159	1.3684E+16	8.0245E+14	2.0951E+13

Also successfully benchmarked the TD nuclear code w/ 130,098 4-component qpwf on 32^3 to prepare for 32x32x50 LACM run with 83,260 4-component qpwfs.

Numerical Studies of Large Amplitude Collective Motion in Nuclear Systems

280Cf responds to quadropole excitation (visit aurel's talk):



Effective Use of the Biggest Open Science Supercomputer in the World Today ... What Comes NEXT???

- finish FY10 Joule OMB PART exercise
- rewrite nuclear codes entirely in C
- further parallelization
  - expose the u and v components of the qpwfs
    - task oriented threading , streaming vectorized regions to GPU
      - not likely to get performance through compiler directives alone
- more i/o testing (if Lustre goes away, then what?)
  - don't see this coming but ... should at least support GPFS
- wide area data movement

Thank You.