Computing Time Dependent Density Functional Theory for Superfluid Systems

A. Bulgac , P. Magierski , K. Roche*

*This research is sponsored by the Office of Advanced Scientific Computing Research; U.S. Department of Energy. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. De-AC05-00OR22725.

Theory

- thousands of nuclei , tens of interactions (?)
- 10K to 100K time steps per nucleus
- multiple densities per time step
    - bilinear products of complex four-component single quasiparticle wavefunctions, gradients and time derivatives
- evaluate observables and spectrum

$$Q(\omega) = \sum_{\sigma} \int d^3 r\, dt\, Q(\vec{r}, \sigma, t)\, \rho(\vec{r}, \sigma, t) \exp(i\omega t)$$

Implementation

- Fortran 90 , C
  - GNU Compiler Collection, Portland Group , Intel
- Discretization (MPI dependence)
  - 3D coordinate and momentum lattice representation
    - quasiparticles represented in plane wave basis
    - canonical index distribution algorithm to map quasiparticles to processes
  - Adams-Bashforth-Milne time stencil , O(h^5)
    - expensive but preserves variance in number density and total energy to 1.e-6 over thousands of time steps
- Major Operations per time step (FFTW dependence)
  - (Local) gradients, time derivatives, potential evaluations, (Global) reductions

Observed Scaling Features

- Strong scaling examples (fix problem complexity, add factor of k processes, reduce runtime by factor of k)

Nx=Ny=Nz=30

PEs (jaguarcnl)	576	1152	1728	2304
<NWF / PE>	48	24	16	12
[s]/ts	56.2	28.8	19.3	14.92 (14.05)

Nx=Ny=Nz=40

PEs (jaguar)	942 (x4)	1884(x4)	2826(x4)	3768(x4)
<NWF/PE>	70	36	24	17
[s] / 10 time steps	296.15	153.31	103.17	77.02 (74.03)
Total INS	7.87635E+14	8.15353E+14	8.15732E+14	8.10577E+14
max(INS(PE))	8.38278E+11	4.25906E+11	2.83689E+11	2.13813E+11
Total FLOP	1.84227E+14	1.84997E+14	1.85766E+14	1.86536E+14
max(INS(FLOP))	1.9578E+11	99655061696	66696994580	50218692684

Observed Scaling Features

- Weak scaling example (scale problem complexity by factor k, scale processes by factor k, fix runtime)

N^3	30^3	40^3	50^3
quasiparticles	28288	66796	130528
PEs	168(x4)	942(x4)	3626(x4)
[s] / 10 time steps	297.31	296.15	319.03
Total INS	1.41538E+14	7.87635E+14	3.13385E+15
Total FLOP	3.3787E+13	1.84227E+14	8.22772E+14
Total BYTES	5.37701E+11	3.00957E+12	1.14865E+13

t(50^3)/t(40^3) = 1.077	PE/PE=3.849	INS/INS=3.97	FLOP/FLOP=4.46
t(50^3)/t(30^3) = 1.073	PE/PE=21.583	INS/INS=22.14	FLOP/FLOP=24.35
t(40^3)/t(30^3) = .996	PE/PE=5.607	INS/INS=5.564	FLOP/FLOP=5.45

Memory discussion

N^3	Quasiparticles	BYTEs	BYTEs (OOC)
30^3	28288	5.37701E+11	44280000
40^3	66796	3.00957E+12	104960000
50^3	130528	1.14865E+13	205000000
60^3	226156	3.43902E+13	354240000
70^3	359056	8.6702E+13	562520000
80^3	535516	1.93026E+14	839680000
90^3	763824	3.92007E+14	1195560000
100^3	1046604	7.36809E+14	1640000000
110^3	1393008	1.30528E+15	2182840000
120^3	1808172	2.19966E+15	2833920000
130^3	2299056	3.55592E+15	3603080000

threaded , buffered swapping

touch_kfil();
kfil_wr();
kfil_rd();
delete_kfil();
kwr_seek_r();
krd_seek_r();

These routines are quite general and have some nice features .

Scalable Data Structures using Disk and Network Memory (MPI) for Arbitrary Rank Object Storage and Manipulation

```
8 100 100 400 400
/tmp/525740.dat.0 0           33554431
/tmp/525740.dat.1  33554432    67108863
/tmp/525740.dat.2  67108864    100663295
/tmp/525740.dat.3  100663296 134217727
/tmp/525740.dat.4  134217728   167772159
/tmp/525740.dat.5  167772160  201326591
/tmp/525740.dat.6  201326592  234881023
...
/tmp/525740.dat.377 12650020864  12683575295
/tmp/525740.dat.378 12683575296    12717129727
/tmp/525740.dat.379 12717129728     12750684159
/tmp/525740.dat.380 12750684160  12784238591
/tmp/525740.dat.381 12784238592    12799999999
```

Arbitrary Rank Objects

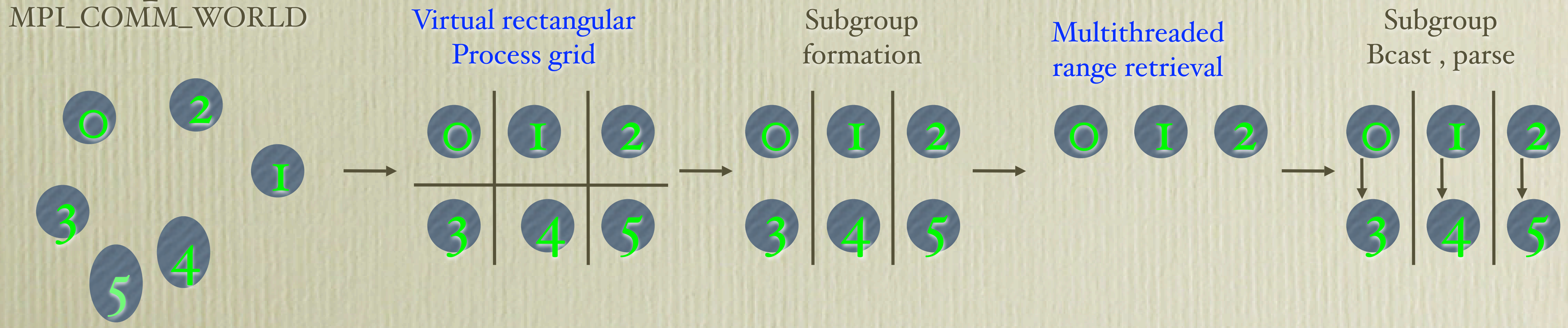
Canonical Mapping :: (e.g. rank-4 object)

$(i0,i1,i2,i3) \dashrightarrow k$

$k = i0 + dim[0] (i1 + dim[1] (i2 + dim[2] (i3)))$

i0 --> inner most loop (fastest varying)
i3 --> outer most loop (slowest)

Example Use: kfil_2dbc_rd() , kfil_2dbc_wr()


MPI_COMM_WORLD
Virtual rectangular Process grid
Subgroup formation
Multithreaded range retrieval
Subgroup Bcast , parse
0
1
2
3
4
5

*replace PATH w/ struct holding MPI id and ptr

- Tasks for years 2 , 3
    - finish ooc hybrid code for enhanced scalability
    - parallelize dft solver , begin coupling to time dependent code ,write test code that diagonalizes for comparison
    - implement Aurel's theoretical changes to homogeneous unitary fermi gas code
    - start implementation of data storage of computed densities and analysis tools for these data sets
- Contributions this year
    - fully parallel td-slda code with nice numerical conservation of critical observables as function of time - as well as exceptional scaling features
    - full support for threaded, asynchronous io for arbitrary rank objects of arbitrary size