

Coupled Cluster theory

Gustav R. Jansen

Department of Physics and Center of Mathematics for Applications
University of Oslo, N-0316 Oslo

27th October 2011

Topics for today.

Choosing a modelspace.

Memory optimization

Compiler optimization

Other types of optimization

Topics for today.

Choosing a modelspace.

Memory optimization

Compiler optimization

Other types of optimization

Choosing a basis

- Harmonic oscillator basis.
- Spherical coordinate system.
- Orbital and intrinsic spin coupled to a total angular momentum \mathbf{j} .
- Defined by quantumnumbers n, l, s, j, m for each type of particle.
- Organized in shells, defined by $N = 2n + l$.

Modelspace

Single particle states

Number of shells	1	2	3	4	5
Number of states	4	16	40	80	140

Number of shells	6	7	8	9	10
Number of states	224	336	480	660	880

Number of shells	11	12	13	14	15
Number of states	1144	1456	1820	2240	2720

We are going to look at modelspaces with 3, 4 and 5 shells, with 1 and 2 shells as the cores.

Modelspace

Modelspace

Core	^4He			^{16}O		
Shells	3	4	5	3	4	5
States	40	80	140	40	80	140
Hole states	4	4	4	16	16	16
Particle states	36	76	136	24	64	124

Memory requirements - ^4He in 3 shells (4,36).

```
$ python memsize.py 4 36
```

Name	Bytes	KBytes	MBytes	GBytes
v:	20480000	20000	19.53	0.0191
f:	12800	12	0.01	0.0000
t1:	1152	1	0.00	0.0000
t1_old:	1152	1	0.00	0.0000
t2:	165888	162	0.16	0.0002
t2_old:	165888	162	0.16	0.0002
=====				
Subtotal:	20826880	20338	19.86	0.0194
=====				
H1:	1152	1	0.00	0.0000
I02a:	10368	10	0.01	0.0000
H2:	10368	10	0.01	0.0000
H3:	128	0	0.00	0.0000
H5:	165888	162	0.16	0.0002
I07a:	18432	18	0.02	0.0000
H7:	18432	18	0.02	0.0000
I10a:	165888	162	0.16	0.0002
I10b:	165888	162	0.16	0.0002
I10c:	165888	162	0.16	0.0002
H10:	165888	162	0.16	0.0002
I11a:	1492992	1458	1.42	0.0014
I12a:	18432	18	0.02	0.0000
H12:	18432	18	0.02	0.0000
=====				
Subtotal:	2418176	2361	2.31	0.0023
=====				
Total:	23245056	22700	22.17	0.0216
=====				

Memory requirements - ^4He in 4 shells (4,76).

```
$ python memsize.py 4 76
```

Name	Bytes	KBytes	MBytes	GBytes
v:	327680000	320000	312.50	0.3052
f:	51200	50	0.05	0.0000
t1:	2432	2	0.00	0.0000
t1_old:	2432	2	0.00	0.0000
t2:	739328	722	0.71	0.0007
t2_old:	739328	722	0.71	0.0007
=====				
Subtotal:	329214720	321498	313.96	0.3066
=====				
H1:	2432	2	0.00	0.0000
I02a:	46208	45	0.04	0.0000
H2:	46208	45	0.04	0.0000
H3:	128	0	0.00	0.0000
H5:	739328	722	0.71	0.0007
I07a:	38912	38	0.04	0.0000
H7:	38912	38	0.04	0.0000
I10a:	739328	722	0.71	0.0007
I10b:	739328	722	0.71	0.0007
I10c:	739328	722	0.71	0.0007
H10:	739328	722	0.71	0.0007
I11a:	14047232	13718	13.40	0.0131
I12a:	38912	38	0.04	0.0000
H12:	38912	38	0.04	0.0000
=====				
Subtotal:	17994496	17572	17.16	0.0168
=====				
Total:	347209216	339071	331.12	0.3234
=====				

Memory requirements - ^4He in 5 shells (4,136).

```
$ python memsize.py 4 136
```

Name	Bytes	KBytes	MBytes	GBytes
v:	3073280000	3001250	2930.91	2.8622
f:	156800	153	0.15	0.0001
t1:	4352	4	0.00	0.0000
t1_old:	4352	4	0.00	0.0000
t2:	2367488	2312	2.26	0.0022
t2_old:	2367488	2312	2.26	0.0022
=====				
Subtotal:	3078180480	3006035	2935.58	2.8668
=====				
H1:	4352	4	0.00	0.0000
I02a:	147968	144	0.14	0.0001
H2:	147968	144	0.14	0.0001
H3:	128	0	0.00	0.0000
H5:	2367488	2312	2.26	0.0022
I07a:	69632	68	0.07	0.0001
H7:	69632	68	0.07	0.0001
I10a:	2367488	2312	2.26	0.0022
I10b:	2367488	2312	2.26	0.0022
I10c:	2367488	2312	2.26	0.0022
H10:	2367488	2312	2.26	0.0022
I11a:	80494592	78608	76.77	0.0750
I12a:	69632	68	0.07	0.0001
H12:	69632	68	0.07	0.0001
=====				
Subtotal:	92910976	90733	88.61	0.0865
=====				
Total:	3171091456	3096769	3024.19	2.9533
=====				

Memory requirements - ^4He in 6 shells (4,220).

```
$ python memsize.py 4 220
```

Name	Bytes	KBytes	MBytes	GBytes
v:	20141047808	19668992	19208.00	18.7578
f:	401408	392	0.38	0.0004
t1:	7040	6	0.01	0.0000
t1_old:	7040	6	0.01	0.0000
t2:	6195200	6050	5.91	0.0058
t2_old:	6195200	6050	5.91	0.0058
=====				
Subtotal:	20153853696	19681497	19220.21	18.7697
=====				
H1:	7040	6	0.01	0.0000
I02a:	387200	378	0.37	0.0004
H2:	387200	378	0.37	0.0004
H3:	128	0	0.00	0.0000
H5:	6195200	6050	5.91	0.0058
I07a:	112640	110	0.11	0.0001
H7:	112640	110	0.11	0.0001
I10a:	6195200	6050	5.91	0.0058
I10b:	6195200	6050	5.91	0.0058
I10c:	6195200	6050	5.91	0.0058
H10:	6195200	6050	5.91	0.0058
I11a:	340736000	332750	324.95	0.3173
I12a:	112640	110	0.11	0.0001
H12:	112640	110	0.11	0.0001
=====				
Subtotal:	372944128	364203	355.67	0.3473
=====				
Total:	20526797824	20045701	19575.88	19.1171
=====				

Memory requirements - ^{16}O in 3 shells (16,24).

```
$ python memsize.py 16 24
```

Name	Bytes	KBytes	MBytes	GBytes
v:	20480000	20000	19.53	0.0191
f:	12800	12	0.01	0.0000
t1:	3072	3	0.00	0.0000
t1_old:	3072	3	0.00	0.0000
t2:	1179648	1152	1.12	0.0011
t2_old:	1179648	1152	1.12	0.0011
=====				
Subtotal:	22858240	22322	21.80	0.0213
=====				
H1:	3072	3	0.00	0.0000
I02a:	4608	4	0.00	0.0000
H2:	4608	4	0.00	0.0000
H3:	2048	2	0.00	0.0000
H5:	1179648	1152	1.12	0.0011
I07a:	786432	768	0.75	0.0007
H7:	786432	768	0.75	0.0007
I10a:	1179648	1152	1.12	0.0011
I10b:	1179648	1152	1.12	0.0011
I10c:	1179648	1152	1.12	0.0011
H10:	1179648	1152	1.12	0.0011
I11a:	1769472	1728	1.69	0.0016
I12a:	786432	768	0.75	0.0007
H12:	786432	768	0.75	0.0007
=====				
Subtotal:	10827776	10574	10.33	0.0101
=====				
Total:	33686016	32896	32.13	0.0314
=====				

Memory requirements - ^{16}O in 4 shells (16,64).

```
$ python memsize.py 16 64
```

Name	Bytes	KBytes	MBytes	GBytes
v:	327680000	320000	312.50	0.3052
f:	51200	50	0.05	0.0000
t1:	8192	8	0.01	0.0000
t1_old:	8192	8	0.01	0.0000
t2:	8388608	8192	8.00	0.0078
t2_old:	8388608	8192	8.00	0.0078
=====				
Subtotal:	344524800	336450	328.56	0.3209
=====				
H1:	8192	8	0.01	0.0000
I02a:	32768	32	0.03	0.0000
H2:	32768	32	0.03	0.0000
H3:	2048	2	0.00	0.0000
H5:	8388608	8192	8.00	0.0078
I07a:	2097152	2048	2.00	0.0020
H7:	2097152	2048	2.00	0.0020
I10a:	8388608	8192	8.00	0.0078
I10b:	8388608	8192	8.00	0.0078
I10c:	8388608	8192	8.00	0.0078
H10:	8388608	8192	8.00	0.0078
I11a:	33554432	32768	32.00	0.0312
I12a:	2097152	2048	2.00	0.0020
H12:	2097152	2048	2.00	0.0020
=====				
Subtotal:	83961856	81994	80.07	0.0782
=====				
Total:	428486656	418444	408.64	0.3991
=====				

Memory requirements - ^{16}O in 5 shells (16,124).

```
$ python memsize.py 16 124
```

Name	Bytes	KBytes	MBytes	GBytes
v:	3073280000	3001250	2930.91	2.8622
f:	156800	153	0.15	0.0001
t1:	15872	15	0.02	0.0000
t1_old:	15872	15	0.02	0.0000
t2:	31490048	30752	30.03	0.0293
t2_old:	31490048	30752	30.03	0.0293
=====				
Subtotal:	3136448640	3062938	2991.15	2.9210
=====				
H1:	15872	15	0.02	0.0000
I02a:	123008	120	0.12	0.0001
H2:	123008	120	0.12	0.0001
H3:	2048	2	0.00	0.0000
H5:	31490048	30752	30.03	0.0293
I07a:	4063232	3968	3.88	0.0038
H7:	4063232	3968	3.88	0.0038
I10a:	31490048	30752	30.03	0.0293
I10b:	31490048	30752	30.03	0.0293
I10c:	31490048	30752	30.03	0.0293
H10:	31490048	30752	30.03	0.0293
I11a:	244047872	238328	232.74	0.2273
I12a:	4063232	3968	3.88	0.0038
H12:	4063232	3968	3.88	0.0038
=====				
Subtotal:	418014976	408217	398.65	0.3893
=====				
Total:	3554463616	3471155	3389.80	3.3104
=====				

Memory requirements - ^{16}O in 6 shells (16,208).

```
$ python memsize.py 16 208
```

Name	Bytes	KBytes	MBytes	GBytes
v:	20141047808	19668992	19208.00	18.7578
f:	401408	392	0.38	0.0004
t1:	26624	26	0.03	0.0000
t1_old:	26624	26	0.03	0.0000
t2:	88604672	86528	84.50	0.0825
t2_old:	88604672	86528	84.50	0.0825
=====				
Subtotal:	20318711808	19842492	19377.43	18.9233
=====				
H1:	26624	26	0.03	0.0000
I02a:	346112	338	0.33	0.0003
H2:	346112	338	0.33	0.0003
H3:	2048	2	0.00	0.0000
H5:	88604672	86528	84.50	0.0825
I07a:	6815744	6656	6.50	0.0063
H7:	6815744	6656	6.50	0.0063
I10a:	88604672	86528	84.50	0.0825
I10b:	88604672	86528	84.50	0.0825
I10c:	88604672	86528	84.50	0.0825
H10:	88604672	86528	84.50	0.0825
I11a:	1151860736	1124864	1098.50	1.0728
I12a:	6815744	6656	6.50	0.0063
H12:	6815744	6656	6.50	0.0063
=====				
Subtotal:	1622867968	1584832	1547.69	1.5114
=====				
Total:	21941579776	21427324	20925.12	20.4347
=====				

Topics for today.

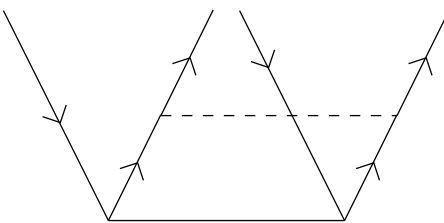
Choosing a modelspace.

Memory optimization

Compiler optimization

Other types of optimization

Example setup

$$t_{ij}^{ab} \Leftarrow \text{diagram} = \frac{1}{2} \langle ab || ef \rangle t_{ij}^{ef}$$


Example 1 - Loops optimized for the result array.

Diagram code.

```
void diagram1(double **** result, double**** v, double***** t2, int nh, int np) {  
    double temp;  
  
    for (int i = 0; i < nh; i++) {  
        for (int j = 0; j < nh; j++) {  
            for (int a = 0; a < np; a++) {  
                for (int b = 0; b < np; b++) {  
                    temp = 0.0;  
                    for (int f = 0; f < np; f++) {  
                        for (int e = 0; e < np; e++) {  
                            temp += v[e+nh][f+nh][a+nh][b+nh] * t2[i][j][e][f];  
                        }  
                    }  
                    result[i][j][a][b] = 0.5*temp;  
                }  
            }  
        }  
    }  
}
```

Memory traversed in strides for both v and t2.

Example 1 - Loops optimized for the result array.

Timed runs.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.4 s	52.18 s	1283.85 s	4.02 s	421.84 s	14273.8 s

Example 2 - Loops optimized for the t2 array.

Diagram code.

```
void diagram2(double **** result, double**** v, double**** t2, int nh, int np) {  
    double temp;  
  
    for (int a = 0; a < np; a++) {  
        for (int b = 0; b < np; b++) {  
            for (int i = 0; i < nh; i++) {  
                for (int j = 0; j < nh; j++) {  
                    temp = 0.0;  
                    for (int e = 0; e < np; e++) {  
                        for (int f = 0; f < np; f++) {  
                            temp += v[e+nh][f+nh][a+nh][b+nh] * t2[i][j][e][f];  
                        }  
                    }  
                    result[i][j][a][b] = 0.5*temp;  
                }  
            }  
        }  
    }  
}
```

Memory traversed in sequence for t2, but not for v.

Example 2 - Loops optimized for the t2 array.

Timed runs.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.4 s	52.18 s	1283.85 s	4.02 s	421.84 s	14273.8 s
Example 2	1.16 s	50.5 s	1282.85 s	3.23 s	408.79 s	14276.4 s

Example 2 - Loops optimized for the t2 array.

Performance counters for example 1 (^{16}O in 5 shells).

14263935.200375	task-clock-msecs	#	0.998 CPUs
17107	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
782571	page-faults	#	0.000 M/sec
31341451383840	cycles	#	2197.251 M/sec
3457587797536	instructions	#	0.110 IPC
384511441380	cache-references	#	26.957 M/sec
61147492	cache-misses	#	0.004 M/sec

Performance counters for example 2 (^{16}O in 5 shells).

14266013.536412	task-clock-msecs	#	0.998 CPUs
15234	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
782571	page-faults	#	0.000 M/sec
31345814514495	cycles	#	2197.237 M/sec
3457593266410	instructions	#	0.110 IPC
384161685589	cache-references	#	26.928 M/sec
20624529	cache-misses	#	0.001 M/sec

Example 3 - Loops optimized for the interaction array.

Diagram code.

```
void diagram3(double **** result, double**** v, double**** t2, int nh, int np) {  
    double temp;  
  
    for (int j = 0; j < nh; j++) {  
        for (int i = 0; i < nh; i++) {  
            for (int e = 0; e < np; e++) {  
                for (int f = 0; f < np; f++) {  
                    for (int a = 0; a < np; a++) {  
                        for (int b = 0; b < np; b++) {  
                            result[i][j][a][b] += 0.5*v[e+nh][f+nh][a+nh][b+nh] * t2[i][j][e][f];  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

Memory traversed in sequence for v, but not for t2.

Example 3 - Loops optimized for the interaction array.

Timed runs.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.4 s	52.18 s	1283.85 s	4.02 s	421.84 s	14273.8 s
Example 2	1.16 s	50.5 s	1282.85 s	3.23 s	408.79 s	14276.4 s
Example 3	0.75 s	14.45 s	146.48 s	2.71 s	124.28 s	1669.25 s

Example 3 - Loops optimized for the interaction array.

Performance counters for example 1 (^{16}O in 5 shells).

14263935.200375	task-clock-msecs	#	0.998 CPUs
17107	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
782571	page-faults	#	0.000 M/sec
31341451383840	cycles	#	2197.251 M/sec
3457587797536	instructions	#	0.110 IPC
384511441380	cache-references	#	26.957 M/sec
61147492	cache-misses	#	0.004 M/sec

Performance counters for example 3 (^{16}O in 5 shells).

1681851.850068	task-clock-msecs	#	0.998 CPUs
2756	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
782571	page-faults	#	0.000 M/sec
3695449350841	cycles	#	2197.250 M/sec
5855460285325	instructions	#	1.585 IPC
619423642974	cache-references	#	368.299 M/sec
4038456	cache-misses	#	0.002 M/sec

Example 4 - Optimized for the interaction and t2 arrays.

Diagram code.

```
void diagram4(double **** result, double**** v, double**** t2, int nh, int np) {  
    double temp;  
    int i;  
  
    for (i = 0; i < nh; i++) {  
        for (int j = 0; j < nh; j++) {  
            for (int a = 0; a < np; a++) {  
                for (int b = 0; b < np; b++) {  
                    temp = 0.0;  
                    for (int e = 0; e < np; e++) {  
                        for (int f = 0; f < np; f++) {  
                            temp += v[a+nh][b+nh][e+nh][f+nh] * t2[i][j][e][f];  
                        }  
                    }  
                    result[i][j][a][b] = 0.5*temp;  
                }  
            }  
        }  
    }  
}
```

Memory traversed in sequence.

Example 4 - Optimized for the interaction and t2 arrays.

Timed runs.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.4 s	52.18 s	1283.85 s	4.02 s	421.84 s	14273.8 s
Example 2	1.16 s	50.5 s	1282.85 s	3.23 s	408.79 s	14276.4 s
Example 3	0.75 s	14.45 s	146.48 s	2.71 s	124.28 s	1669.25 s
Example 4	0.45 s	8.72 s	88.85 s	1.78 s	75.59 s	1024.83 s

Example 4 - Optimized for the interaction and t2 arrays.

Performance counters for example 3 (^{16}O in 5 shells).

1681851.850068	task-clock-msecs	#	0.998 CPUs
2756	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
782571	page-faults	#	0.000 M/sec
3695449350841	cycles	#	2197.250 M/sec
5855460285325	instructions	#	1.585 IPC
619423642974	cache-references	#	368.299 M/sec
4038456	cache-misses	#	0.002 M/sec

Performance counters for example 4 (^{16}O in 5 shells).

1038379.096891	task-clock-msecs	#	0.998 CPUs
1692	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
782571	page-faults	#	0.001 M/sec
2281578423505	cycles	#	2197.250 M/sec
3433226495257	instructions	#	1.505 IPC
437426640824	cache-references	#	421.259 M/sec
3514505	cache-misses	#	0.003 M/sec

Example 5 - Interaction and t2 packed into matrices.

Diagram code.

```
void diagram5(double ** result, double** v, double** t2, int nh, int np) {  
    double temp;  
  
    for (int ij = 0; ij < nh; ij++) {  
        for (int ab = 0; ab < np; ab++) {  
            temp = 0.0;  
            for (int ef = 0; ef < np; ef++) {  
                temp += v[ab+nh][ef+nh] * t2[ij][ef];  
            }  
            result[ij][ab] = 0.5*temp;  
        }  
    }  
}
```

Memory traversed in sequence, one extra lookup for each element.

Example 5 - Interaction and t2 packed into matrices.

Timed runs.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.4 s	52.18 s	1283.85 s	4.02 s	421.84 s	14273.8 s
Example 2	1.16 s	50.5 s	1282.85 s	3.23 s	408.79 s	14276.4 s
Example 3	0.75 s	14.45 s	146.48 s	2.71 s	124.28 s	1669.25 s
Example 4	0.45 s	8.72 s	88.85 s	1.78 s	75.59 s	1024.83 s
Example 5	0.28 s	5.45 s	55.66 s	0.87 s	43.67 s	616.01 s

Example 5 - Interaction and t2 packed into matrices.

Performance counters for example 4 (^{16}O in 5 shells).

1038379.096891	task-clock-msecs	#	0.998 CPUs
1692	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
782571	page-faults	#	0.001 M/sec
2281578423505	cycles	#	2197.250 M/sec
3433226495257	instructions	#	1.505 IPC
437426640824	cache-references	#	421.259 M/sec
3514505	cache-misses	#	0.003 M/sec

Performance counters for example 5 (^{16}O in 5 shells).

629105.981266	task-clock-msecs	#	0.998 CPUs
652	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
780132	page-faults	#	0.001 M/sec
1382287129886	cycles	#	2197.225 M/sec
1970719597193	instructions	#	1.426 IPC
248072718967	cache-references	#	394.326 M/sec
3646415	cache-misses	#	0.006 M/sec

Example 6 - Split interaction and t2 packed into matrices.

Diagram code.

```
void diagram6(double ** result, double** v, double** t2, int nh, int np) {  
    double temp;  
  
    for (int ij = 0; ij < nh; ij++) {  
        for (int ab = 0; ab < np; ab++) {  
            temp = 0.0;  
            for (int ef = 0; ef < np; ef++) {  
                temp += v[ab][ef] * t2[ij][ef];  
            }  
            result[ij][ab] = 0.5*temp;  
        }  
    }  
}
```

Memory traversed in sequence, one extra lookup for each element.

Example 6 - Split interaction and t2 packed into matrices.

Timed runs.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.4 s	52.18 s	1283.85 s	4.02 s	421.84 s	14273.8 s
Example 2	1.16 s	50.5 s	1282.85 s	3.23 s	408.79 s	14276.4 s
Example 3	0.75 s	14.45 s	146.48 s	2.71 s	124.28 s	1669.25 s
Example 4	0.45 s	8.72 s	88.85 s	1.78 s	75.59 s	1024.83 s
Example 5	0.28 s	5.45 s	55.66 s	0.87 s	43.67 s	616.01 s
Example 6	0.25 s	4.95 s	51.18 s	0.78 s	41.39 s	563.26 s

Example 6 - Split interaction and t2 packed into matrices.

Performance counters for example 5 (^{16}O in 5 shells).

629105.981266	task-clock-msecs	#	0.998 CPUs
652	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
780132	page-faults	#	0.001 M/sec
1382287129886	cycles	#	2197.225 M/sec
1970719597193	instructions	#	1.426 IPC
248072718967	cache-references	#	394.326 M/sec
3646415	cache-misses	#	0.006 M/sec

Performance counters for example 6 (^{16}O in 5 shells).

576272.052952	task-clock-msecs	#	0.998 CPUs
947	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
780132	page-faults	#	0.001 M/sec
1266203387910	cycles	#	2197.232 M/sec
1728679956955	instructions	#	1.365 IPC
248065776595	cache-references	#	430.466 M/sec
5702346	cache-misses	#	0.010 M/sec

Example 7 - Interaction and t2 packed into arrays.

Diagram code.

```
void diagram7(double *result, double *v, double *t2, int nh, int np) {  
    double temp;  
    int vi=0, ti=0, ri=0, ti1 = 0;  
  
    for (int i = 0; i < nh; i++) {  
        for (int j = 0; j < nh; j++) {  
            vi = 0;  
            for (int a = 0; a < np; a++) {  
                for (int b = 0; b < np; b++) {  
                    ti = ti1; temp = 0.0;  
                    for (int e = 0; e < np; e++) {  
                        for (int f = 0; f < np; f++) {  
                            temp += v[vi] * t2[ti];  
                            vi += 1; ti += 1;  
                        }  
                    }  
                    result[ri] = 0.5*temp;  
                    ri += 1;  
                }  
            }  
            ti1 = ti;  
        }  
    }  
}
```

Memory traversed in sequence, no extra lookups.

Example 7 - Interaction and t2 packed into arrays.

Timed runs.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.4 s	52.18 s	1283.85 s	4.02 s	421.84 s	14273.8 s
Example 2	1.16 s	50.5 s	1282.85 s	3.23 s	408.79 s	14276.4 s
Example 3	0.75 s	14.45 s	146.48 s	2.71 s	124.28 s	1669.25 s
Example 4	0.45 s	8.72 s	88.85 s	1.78 s	75.59 s	1024.83 s
Example 5	0.28 s	5.45 s	55.66 s	0.87 s	43.67 s	616.01 s
Example 6	0.25 s	4.95 s	51.18 s	0.78 s	41.39 s	563.26 s
Example 7	0.24 s	4.88 s	50.13 s	0.78 s	38.73 s	569.37 s

Example 7 - Interaction and t2 packed into arrays.

Performance counters for example 6 (^{16}O in 5 shells).

576272.052952	task-clock-msecs	#	0.998 CPUs
947	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
780132	page-faults	#	0.001 M/sec
1266203387910	cycles	#	2197.232 M/sec
1728679956955	instructions	#	1.365 IPC
248065776595	cache-references	#	430.466 M/sec
5702346	cache-misses	#	0.010 M/sec

Performance counters for example 7 (^{16}O in 5 shells).

577006.624301	task-clock-msecs	#	0.998 CPUs
579	context-switches	#	0.000 M/sec
0	CPU-migrations	#	0.000 M/sec
477453	page-faults	#	0.001 M/sec
1267825233931	cycles	#	2197.246 M/sec
1235236115940	instructions	#	0.974 IPC
192456007395	cache-references	#	333.542 M/sec
3383812	cache-misses	#	0.006 M/sec

Topics for today.

Choosing a modelspace.

Memory optimization

Compiler optimization

Other types of optimization

Compiler based optimization.

- Know your compiler!
- What optimizations are available in my compiler?
- Are there any dangerous areas for my compiler?
- Can I use a different compiler?
- Am I using the correct libraries?

All examples optimized using -O1.

Timed runs - original.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.4 s	52.18 s	1283.85 s	4.02 s	421.84 s	14273.8 s
Example 2	1.16 s	50.5 s	1282.85 s	3.23 s	408.79 s	14276.4 s
Example 3	0.75 s	14.45 s	146.48 s	2.71 s	124.28 s	1669.25 s
Example 4	0.45 s	8.72 s	88.85 s	1.78 s	75.59 s	1024.83 s
Example 5	0.28 s	5.45 s	55.66 s	0.87 s	43.67 s	616.01 s
Example 6	0.25 s	4.95 s	51.18 s	0.78 s	41.39 s	563.26 s
Example 7	0.24 s	4.88 s	50.13 s	0.78 s	38.73 s	569.37 s

Timed runs for -O1.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.15 s	50.03 s	1181.01 s	3.05 s	405.39 s	13168.9 s
Example 2	1.13 s	49.79 s	1181.73 s	2.96 s	403.99 s	13231.7 s
Example 3	0.28 s	5.46 s	55.60 s	1.18 s	48.64 s	650.18 s
Example 4	0.09 s	1.51 s	14.56 s	0.43 s	14.28 s	176.55 s
Example 5	0.06 s	1.34 s	14.38 s	0.21 s	10.69 s	150.72 s
Example 6	0.07 s	1.26 s	13.64 s	0.20 s	10.13 s	142.31 s
Example 7	0.08 s	1.51 s	15.30 s	0.24 s	12.34 s	169.35 s

All examples optimized using -O2.

Timed runs for -O1.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.15 s	50.03 s	1181.01 s	3.05 s	405.39 s	13168.9 s
Example 2	1.13 s	49.79 s	1181.73 s	2.96 s	403.99 s	13231.7 s
Example 3	0.28 s	5.46 s	55.60 s	1.18 s	48.64 s	650.18 s
Example 4	0.09 s	1.51 s	14.56 s	0.43 s	14.28 s	176.55 s
Example 5	0.06 s	1.34 s	14.38 s	0.21 s	10.69 s	150.72 s
Example 6	0.07 s	1.26 s	13.64 s	0.20 s	10.13 s	142.31 s
Example 7	0.08 s	1.51 s	15.30 s	0.24 s	12.34 s	169.35 s

Timed runs for -O2.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.15 s	50.01 s	1181.97 s	3.04 s	404.67 s	13449.6 s
Example 2	1.14 s	49.78 s	1238.09 s	2.97 s	404.25 s	14206.8 s
Example 3	0.10 s	1.67 s	16.39 s	0.54 s	16.85 s	206.17 s
Example 4	0.09 s	1.50 s	14.64 s	0.44 s	14.42 s	178.33 s
Example 5	0.07 s	1.40 s	15.96 s	0.23 s	11.27 s	158.48 s
Example 6	0.08 s	1.39 s	14.93 s	0.22 s	11.09 s	156.52 s
Example 7	0.07 s	1.42 s	14.16 s	0.23 s	11.75 s	157.58 s

All examples optimized using -O3.

Timed runs for -O2.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.15 s	50.01 s	1181.97 s	3.04 s	404.67 s	13449.6 s
Example 2	1.14 s	49.78 s	1238.09 s	2.97 s	404.25 s	14206.8 s
Example 3	0.10 s	1.67 s	16.39 s	0.54 s	16.85 s	206.17 s
Example 4	0.09 s	1.50 s	14.64 s	0.44 s	14.42 s	178.33 s
Example 5	0.07 s	1.40 s	15.96 s	0.23 s	11.27 s	158.48 s
Example 6	0.08 s	1.39 s	14.93 s	0.22 s	11.09 s	156.52 s
Example 7	0.07 s	1.42 s	14.16 s	0.23 s	11.75 s	157.58 s

Timed runs for -O3.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.16 s	50.01 s	1180.76 s	3.04 s	405.25 s	13162.6 s
Example 2	1.15 s	49.78 s	1276.90 s	2.97 s	403.89 s	13716.0 s
Example 3	0.10 s	1.68 s	16.39 s	0.54 s	16.75 s	205.27 s
Example 4	0.09 s	1.51 s	14.65 s	0.43 s	14.41 s	178.12 s
Example 5	0.07 s	1.41 s	15.95 s	0.23 s	11.27 s	158.56 s
Example 6	0.07 s	1.39 s	14.91 s	0.23 s	11.09 s	156.42 s
Example 7	0.08 s	1.42 s	14.10 s	0.23 s	11.72 s	156.86 s

Performance gain.

Best/worst cases.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Worst	1.4 s	52.18 s	1283.85 s	4.02 s	421.84 s	14273.8 s
Example	1	1	1	1	1	1
Optimization	None	None	None	None	None	None
Best	0.06 s	1.26 s	13.64 s	0.20 s	10.13 s	142.31 s
Example	5	6	6	6	6	6
Optimization	-O1	-O1	-O1	-O1	-O1	-O1
Gain	23.3	41.4	86.8	21.0	41.64	100.3

Optimization summary

- Avoid memory strides in loops.
- Avoid pointers to pointers to pointers . . .
- For best performance, tweak all diagrams and experiment with compiler settings.
- For acceptable performance, make sure that arrays are traversed in sequence and let the compiler figure out the rest.
- Very often, the compiler can figure out a better optimization of a simple code, than of a complicated and at a first glance, more efficient code.
- For small systems - quick and dirty gets the job done, but for larger systems, more time should be spent looking for performance gains.

Topics for today.

Choosing a modelspace.

Memory optimization

Compiler optimization

Other types of optimization

Physics based optimization

- Symmetries.
- Approximations.

Example 9 - Implementation in jj-scheme

Diagram code.

```

void diagram8(double *result, double *v, double *t2, int nh, int np, int max_j) {
    double temp;
    int vi=0, ti=0, ri=0;

    for (int jj = 1; jj <= max_j; jj++) {
        for (int i = 0; i < nh; i++) {
            for (int j = 0; j < nh; j++) {
                vi = (jj-1)*np*np*np*np;
                for (int a = 0; a < np; a++) {
                    for (int b = 0; b < np; b++) {
                        ti = (jj-1)*nh*nh*np*np + i*nh*np*np + j*np*np;
                        temp = 0.0;
                        for (int e = 0; e < np; e++) {
                            for (int f = 0; f < np; f++) {
                                temp += v[vi] * t2[ti];
                                vi += 1;
                                ti += 1;
                            }
                        }
                        result[ri] = 0.5*temp;
                        ri += 1;
                    }
                }
            }
        }
    }
}

```

Example 9 - Implementation in jj-scheme

Timed runs for -O3.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 1	1.16 s	50.01 s	1180.76 s	3.04 s	405.25 s	13162.6 s
Example 2	1.15 s	49.78 s	1276.90 s	2.97 s	403.89 s	13716.0 s
Example 3	0.10 s	1.68 s	16.39 s	0.54 s	16.75 s	205.27 s
Example 4	0.09 s	1.51 s	14.65 s	0.43 s	14.41 s	178.12 s
Example 5	0.07 s	1.41 s	15.95 s	0.23 s	11.27 s	158.56 s
Example 6	0.07 s	1.39 s	14.91 s	0.23 s	11.09 s	156.42 s
Example 7	0.08 s	1.42 s	14.10 s	0.23 s	11.72 s	156.86 s
Example 9	0.00043 s	0.0086 s	0.069 s	0.00058 s	0.0281 s	0.317 s

Example 9 - Implementation in jj-scheme

Worst case compared to jj-scheme.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Worst	1.4 s	52.18 s	1283.85 s	4.02 s	421.84 s	14273.8 s
jj-scheme	0.00043 s	0.0086 s	0.069 s	0.00058 s	0.0281 s	0.317 s
Gain	3256	6067	18607	6931	15012	45027

One year in worst m-scheme implementation, would take 11 minutes in jj-scheme.

Best case compared to jj-scheme.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Best	0.06 s	1.26 s	13.64 s	0.20 s	10.13 s	142.31 s
jj-scheme	0.00043 s	0.0086 s	0.069 s	0.00058 s	0.0281 s	0.317 s
Gain	140	147	198	345	360	449

One year in best m-scheme implementation, would take 19 hours in jj-scheme.

Example 9 - Implementation in jj-scheme

Timed runs for -O3.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	6	7	8	6	7	8
Example 9	0.338 s	1.27 s	4.37 s	2.09 s	9.04 s	31.28 s

Model space

Single particle states

Number of shells	1	2	3	4	5
m-scheme states	4	16	40	80	140
jj-scheme states	2	6	12	20	30

Number of shells	6	7	8	9	10
m-scheme states	224	336	480	660	880
jj-scheme states	42	56	72	90	110

Number of shells	11	12	13	14	15
m-scheme states	1144	1456	1820	2240	2720
jj-scheme states	132	156	182	210	240

Modelspace

Modelspace

Core	^4He			^{16}O		
Shells	6	7	8	6	7	8
States	42	56	72	42	56	72
Hole states	2	2	2	6	6	6
Particle states	40	54	70	36	50	66

Memory requirements - ^4He in 6 shells (2,40).

```
$ python memsize.py 2 40 jj
```

Name	Bytes	KBytes	MBytes	GBytes
v:	248935680	243101	237.40	0.2318
f:	141120	137	0.13	0.0001
t1:	6400	6	0.01	0.0000
t1_old:	6400	6	0.01	0.0000
t2:	512000	500	0.49	0.0005
t2_old:	512000	500	0.49	0.0005
=====				
Subtotal:	250113600	244251	238.53	0.2329
=====				
H1:	6400	6	0.01	0.0000
I02a:	128000	125	0.12	0.0001
H2:	128000	125	0.12	0.0001
H3:	320	0	0.00	0.0000
H5:	512000	500	0.49	0.0005
I07a:	25600	25	0.02	0.0000
H7:	25600	25	0.02	0.0000
I10a:	512000	500	0.49	0.0005
I10b:	512000	500	0.49	0.0005
I10c:	512000	500	0.49	0.0005
H10:	512000	500	0.49	0.0005
I11a:	10240000	10000	9.77	0.0095
I12a:	25600	25	0.02	0.0000
H12:	25600	25	0.02	0.0000
=====				
Subtotal:	13165120	12856	12.56	0.0123
=====				
Total:	263278720	257108	251.08	0.2452
=====				

Memory requirements - ^4He in 7 shells (2,54).

```
$ python memsize.py 2 54 jj
```

Name	Bytes	KBytes	MBytes	GBytes
v:	786759680	768320	750.31	0.7327
f:	250880	245	0.24	0.0002
t1:	8640	8	0.01	0.0000
t1_old:	8640	8	0.01	0.0000
t2:	933120	911	0.89	0.0009
t2_old:	933120	911	0.89	0.0009
=====				
Subtotal:	788894080	770404	752.35	0.7347
=====				
H1:	8640	8	0.01	0.0000
I02a:	233280	227	0.22	0.0002
H2:	233280	227	0.22	0.0002
H3:	320	0	0.00	0.0000
H5:	933120	911	0.89	0.0009
I07a:	34560	33	0.03	0.0000
H7:	34560	33	0.03	0.0000
I10a:	933120	911	0.89	0.0009
I10b:	933120	911	0.89	0.0009
I10c:	933120	911	0.89	0.0009
H10:	933120	911	0.89	0.0009
I11a:	25194240	24603	24.03	0.0235
I12a:	34560	33	0.03	0.0000
H12:	34560	33	0.03	0.0000
=====				
Subtotal:	30473600	29759	29.06	0.0284
=====				
Total:	819367680	800163	781.41	0.7631
=====				

Memory requirements - ^4He in 8 shells (2,70).

```
$ python memsize.py 2 70 jj
```

Name	Bytes	KBytes	MBytes	GBytes
v:	2149908480	2099520	2050.31	2.0023
f:	414720	405	0.40	0.0004
t1:	11200	10	0.01	0.0000
t1_old:	11200	10	0.01	0.0000
t2:	1568000	1531	1.50	0.0015
t2_old:	1568000	1531	1.50	0.0015
=====				
Subtotal:	2153481600	2103009	2053.72	2.0056
=====				
H1:	11200	10	0.01	0.0000
I02a:	392000	382	0.37	0.0004
H2:	392000	382	0.37	0.0004
H3:	320	0	0.00	0.0000
H5:	1568000	1531	1.50	0.0015
I07a:	44800	43	0.04	0.0000
H7:	44800	43	0.04	0.0000
I10a:	1568000	1531	1.50	0.0015
I10b:	1568000	1531	1.50	0.0015
I10c:	1568000	1531	1.50	0.0015
H10:	1568000	1531	1.50	0.0015
I11a:	54880000	53593	52.34	0.0511
I12a:	44800	43	0.04	0.0000
H12:	44800	43	0.04	0.0000
=====				
Subtotal:	63694720	62201	60.74	0.0593
=====				
Total:	2217176320	2165211	2114.46	2.0649
=====				

Memory requirements - ^4He in 9 shells (2,88).

```
$ python memsize.py 2 88 jj
```

Name	Bytes	KBytes	MBytes	GBytes
v:	5248800000	5125781	5005.65	4.8883
f:	648000	632	0.62	0.0006
t1:	14080	13	0.01	0.0000
t1_old:	14080	13	0.01	0.0000
t2:	2478080	2420	2.36	0.0023
t2_old:	2478080	2420	2.36	0.0023
=====				
Subtotal:	5254432320	5131281	5011.02	4.8936
=====				
H1:	14080	13	0.01	0.0000
I02a:	619520	605	0.59	0.0006
H2:	619520	605	0.59	0.0006
H3:	320	0	0.00	0.0000
H5:	2478080	2420	2.36	0.0023
I07a:	56320	55	0.05	0.0001
H7:	56320	55	0.05	0.0001
I10a:	2478080	2420	2.36	0.0023
I10b:	2478080	2420	2.36	0.0023
I10c:	2478080	2420	2.36	0.0023
H10:	2478080	2420	2.36	0.0023
I11a:	109035520	106480	103.98	0.1015
I12a:	56320	55	0.05	0.0001
H12:	56320	55	0.05	0.0001
=====				
Subtotal:	122904640	120024	117.21	0.1145
=====				
Total:	5377336960	5251305	5128.23	5.0080
=====				

Parallel execution

Openmp

- Shared memory.
- Typical multicore cpu's and multiple cpu's on one motherboard.
- Threads based.
- All instances run under one process.
- Threads are created and terminated as needed inside a process.
- Little or no overhead on simple loops.
- Watch out for libraries that are not threadsafe.

MPI

- Separate memory.
- Multinode clusters or shared memory systems.
- Based on message passing between processes.
- Each instance run under a separate process.
- All processes execute the program from start to finish.
- Noticeable overhead for small systems.

Example 8 - Openmp.

Diagram code.

```
void diagram8(double ** result, double** v, double** t2, int nh, int np) {  
    double temp;  
    int n, th;  
  
    #pragma omp parallel default(shared)  
    n = omp_get_num_threads();  
    th = omp_get_thread_num();  
    {  
        #pragma omp for schedule(static) nowait  
        for (int ij = th; ij < nh; ij+= n) {  
            for (int ab = 0; ab < np; ab++) {  
                temp = 0.0;  
                for (int ef = 0; ef < np; ef++) {  
                    temp += v[ab][ef] * t2[ij][ef];  
                }  
                result[ij][ab] = 0.5*temp;  
            }  
        }  
    }  
}
```

Example 8 - Openmp.

Timed runs - Unoptimized.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 6	0.25 s	4.95 s	51.18 s	0.78 s	41.39 s	563.26 s
Example 8	0.06 s	1.27 s	12.87 s	0.20 s	10.20 s	145.39 s

Timed runs - Compiled with -O1.

Core	^4He - 4 hole states			^{16}O - 16 hole states		
Shells	3	4	5	3	4	5
Example 6	0.07 s	1.26 s	13.64 s	0.20 s	10.13 s	142.31 s
Example 8	0.02 s	0.34 s	3.74 s	0.06 s	2.77 s	39.55 s

Example - MPI.

Diagram code.

```
void diagram10(double ** result, double** v, double** t2, int nh, int np) {  
    double temp;  
    int n, th;  
    MPI_Comm_size(MPI_COMM_WORLD,&n);  
    MPI_Comm_rank(MPI_COMM_WORLD,&th);  
  
    {  
        for (int ij = th; ij < nh; ij+= n) {  
            for (int ab = 0; ab < np; ab++) {  
                temp = 0.0;  
                for (int ef = 0; ef < np; ef++) {  
                    temp += v[ab][ef] * t2[ij][ef];  
                }  
                result[ij][ab] = 0.5*temp;  
            }  
        }  
    }  
}
```

Memory requirements - ^4He in 5 shells (4,136).

```
$ python memsize.py 4 136
```

Name	Bytes	KBytes	MBytes	GBytes
v:	3073280000	3001250	2930.91	2.8622
(vpppp):	(2736816128)	(2672672)	(2610.03)	(2.5513)
f:	156800	153	0.15	0.0001
t1:	4352	4	0.00	0.0000
t1_old:	4352	4	0.00	0.0000
t2:	2367488	2312	2.26	0.0022
t2_old:	2367488	2312	2.26	0.0022
=====				
Subtotal:	3078180480	3006035	2935.58	2.8668
=====				
H1:	4352	4	0.00	0.0000
I02a:	147968	144	0.14	0.0001
H2:	147968	144	0.14	0.0001
H3:	128	0	0.00	0.0000
H5:	2367488	2312	2.26	0.0022
I07a:	69632	68	0.07	0.0001
H7:	69632	68	0.07	0.0001
I10a:	2367488	2312	2.26	0.0022
I10b:	2367488	2312	2.26	0.0022
I10c:	2367488	2312	2.26	0.0022
H10:	2367488	2312	2.26	0.0022
I11a:	80494592	78608	76.77	0.0750
I12a:	69632	68	0.07	0.0001
H12:	69632	68	0.07	0.0001
=====				
Subtotal:	92910976	90733	88.61	0.0865
=====				
Total:	3171091456	3096769	3024.19	2.9533
=====				

Example - MPI.

Diagram code.

```
void diagram10(double ** result, double** v, double** t2, int nh, int np) {  
    double temp;  
    int n, th, ab;  
    MPI_Comm_size(MPI_COMM_WORLD,&n);  
    MPI_Comm_rank(MPI_COMM_WORLD,&th);  
  
    {  
    for (int ij = 0; ij < nh; ij++) {  
        for (int abn = 0; abn < n_lookup[th]; abn++) {  
            temp = 0.0;  
            for (int ef = 0; ef < np; ef++) {  
                temp += v[abn][ef] * t2[ij][ef];  
            }  
            ab = lookup[th][abn]  
            result[ij][ab] = 0.5*temp;  
        }  
    }  
}
```

Memory requirements - ^{16}O in 6 shells (16,208).

```
$ python memsize.py 16 208
```

Name	Bytes	KBytes	MBytes	GBytes
v:	20141047808	19668992	19208.00	18.7578
(vpppp):	(14974189568)	(14623232)	(14280.50)	(13.9458)
f:	401408	392	0.38	0.0004
t1:	26624	26	0.03	0.0000
t1_old:	26624	26	0.03	0.0000
t2:	88604672	86528	84.50	0.0825
t2_old:	88604672	86528	84.50	0.0825
=====				
Subtotal:	20318711808	19842492	19377.43	18.9233
=====				
H1:	26624	26	0.03	0.0000
I02a:	346112	338	0.33	0.0003
H2:	346112	338	0.33	0.0003
H3:	2048	2	0.00	0.0000
H5:	88604672	86528	84.50	0.0825
I07a:	6815744	6656	6.50	0.0063
H7:	6815744	6656	6.50	0.0063
I10a:	88604672	86528	84.50	0.0825
I10b:	88604672	86528	84.50	0.0825
I10c:	88604672	86528	84.50	0.0825
H10:	88604672	86528	84.50	0.0825
I11a:	1151860736	1124864	1098.50	1.0728
I12a:	6815744	6656	6.50	0.0063
H12:	6815744	6656	6.50	0.0063
=====				
Subtotal:	1622867968	1584832	1547.69	1.5114
=====				
Total:	21941579776	21427324	20925.12	20.4347
=====				

Memory requirements - ^{16}O in 7 shells (16,320).

```
$ python memsize.py 16 320
```

Name	Bytes	KBytes	MBytes	GBytes
v:	101964054528	99574272	97240.50	94.9614
f:	903168	882	0.86	0.0008
t1:	40960	40	0.04	0.0000
t1_old:	40960	40	0.04	0.0000
t2:	209715200	204800	200.00	0.1953
t2_old:	209715200	204800	200.00	0.1953
=====				
Subtotal:	102384470016	99984834	97641.44	95.3530
=====				
H1:	40960	40	0.04	0.0000
I02a:	819200	800	0.78	0.0008
H2:	819200	800	0.78	0.0008
H3:	2048	2	0.00	0.0000
H5:	209715200	204800	200.00	0.1953
I07a:	10485760	10240	10.00	0.0098
H7:	10485760	10240	10.00	0.0098
I10a:	209715200	204800	200.00	0.1953
I10b:	209715200	204800	200.00	0.1953
I10c:	209715200	204800	200.00	0.1953
H10:	209715200	204800	200.00	0.1953
I11a:	4194304000	4096000	4000.00	3.9062
I12a:	10485760	10240	10.00	0.0098
H12:	10485760	10240	10.00	0.0098
=====				
Subtotal:	5286504448	5162602	5041.60	4.9234
=====				
Total:	107670974464	105147436	102683.04	100.2764
=====				

Memory requirements - ^{16}O in 10 shells (16,864).

```
$ python memsize.py 16 864
```

Name	Bytes	KBytes	MBytes	GBytes
v:	4797562880000	4685120000	4575312.50	4468.0786
f:	6195200	6050	5.91	0.0058
t1:	110592	108	0.11	0.0001
t1_old:	110592	108	0.11	0.0001
t2:	1528823808	1492992	1458.00	1.4238
t2_old:	1528823808	1492992	1458.00	1.4238
=====				
Subtotal:	4800626944000	4688112250	4578234.62	4470.9322
=====				
H1:	110592	108	0.11	0.0001
I02a:	5971968	5832	5.70	0.0056
H2:	5971968	5832	5.70	0.0056
H3:	2048	2	0.00	0.0000
H5:	1528823808	1492992	1458.00	1.4238
I07a:	28311552	27648	27.00	0.0264
H7:	28311552	27648	27.00	0.0264
I10a:	1528823808	1492992	1458.00	1.4238
I10b:	1528823808	1492992	1458.00	1.4238
I10c:	1528823808	1492992	1458.00	1.4238
H10:	1528823808	1492992	1458.00	1.4238
I11a:	82556485632	80621568	78732.00	76.8867
I12a:	28311552	27648	27.00	0.0264
H12:	28311552	27648	27.00	0.0264
=====				
Subtotal:	90325907456	88208894	86141.50	84.1226
=====				
Total:	4890952851456	4776321144	4664376.12	4555.0548
=====				

Memory requirements - ^{16}O in 15 shells (16,864).

```
$ python memsize.py 16 2704
```

Name	Bytes	KBytes	MBytes	GBytes
v:437890580480000		427627520000	417605000.00	407817.3828
f:	59187200	57800	56.45	0.0551
t1:	346112	338	0.33	0.0003
t1_old:	346112	338	0.33	0.0003
t2:	14974189568	14623232	14280.50	13.9458
t2_old:	14974189568	14623232	14280.50	13.9458
=====				
Subtotal:	437920588738560	427656824940	417633618.11	407845.3302
=====				
H1:	346112	338	0.33	0.0003
I02a:	58492928	57122	55.78	0.0545
H2:	58492928	57122	55.78	0.0545
H3:	2048	2	0.00	0.0000
H5:	14974189568	14623232	14280.50	13.9458
I07a:	88604672	86528	84.50	0.0825
H7:	88604672	86528	84.50	0.0825
I10a:	14974189568	14623232	14280.50	13.9458
I10b:	14974189568	14623232	14280.50	13.9458
I10c:	14974189568	14623232	14280.50	13.9458
H10:	14974189568	14623232	14280.50	13.9458
I11a:	2530638036992	2471326208	2413404.50	2356.8403
I12a:	88604672	86528	84.50	0.0825
H12:	88604672	86528	84.50	0.0825
=====				
Subtotal:	2605980737536	2544903064	2485256.90	2427.0087
=====				
Total:	440526569476096	430201728004	420118875.00	410272.3389
=====				

What now?

- Where to we get 407 Tb of interaction elements? From file?
- Is it possible to get 205136 nodes to work together with MPI?
- Any ideas?