Many Body Algorithms

Andrey Asadchev

Iowa State University

December 7, 2011

Outline

- Last Presentation
- Coupled Cluster
- Programming

Last Presentation

- Many Body methods are memory and CPU hogs
- It is possible to implement algorithms to use $O(N^3)$ without sacrificing efficiency.
- Faster runtime can be achieved using accelerators/clusters.
- Large arrays can be stored in distributed memory or disk.

Einstein notation is a compact way to manipulate tensors:
inner product: $a = x^i y_i$
outer product: $t^i_j = x^i y_j$
matrix product: $t^i_j = a^i_k b^k_j$
transpose: $t^i_j = a^j_i$
integral transformation: $v^{ij}_{kl} = C^i_p C^j_r C^q_k C^s_l G^{pr}_{qs}$

Integrals and Amplitudes

Let $i, j, k, l, \ldots$ refer to O indices
Let $a, b, c, d, \ldots$ refer to V indices
Let $p, q, r, s, \ldots$ be atomic indices A $V >> 0$

- $V^{ij}_{ab} = V^{ji}_{ba}$
- $V^{ia}_{jb} = V^{ja}_{ib} = V^{ib}_{ja}$
- $t^{ij}_{ab} = t^{ji}_{ba}$

VVVV term

VVVV term is too large to store, evaluated on the fly.
Many ways to do so, but one is more interesting:
$V^{ab}_{cd} t^{cd}_{ij} = (V^{ab}_{qs} C^q_c C^s_d) t^{cd}_{ij} = V^{ab}_{qs} (C^q_c C^s_d t^{cd}_{ij}) =$
$V^{ab}_{qs} t^{qs}_{ij} = (C^p_a C^r_b) V^{pr}_{qs} t^{qs}_{ij} =$
$(C^p_a C^r_b) U^{pr}_{ij}$
Why?

- U^{pr}_{ij} symmetry:
  $U^{pr}_{ij} == U^{rp}_{ji}$ for any quartets p, r. 2x fewer computations.
- Other VT terms:
  Most of work is in first two transformations. Third and fourth transformations are cheap. Since last indices are in AO basis, any $VT2_{ij}$ can be formed for free.
- Small memory footprint

U_{ij}^{pr}

- To transform in MO index, entire AO index is needed
- To transform in AO basis, only corresponding AO index is needed
- Therefore, working storage can be as small as the shell quartet, converted as deemed necessary to U_{ij}^{pr} *pr* segment
- Small U_{ij}^{pr} *pr* segment allows to increase *pr* tile, which has a direct effect on how many times t_{ij}^{qs} must be loaded

Listing 1: Direct CC

```
for QS in (S, Q <= S) {
  for R in Basis {
    for (Q,S) in QS {
      for P in Basis {
        V(p,q,s,r) = eri(P,Q,R,S)
        for r in R {
          load t(O,O,P,r)
          U(i,j,qs) += t(i,j,p)*V(p,qs,r)
        }
      }
    }
  }
}
store U(i,j,qs)
store U(j,i,sq)
```

Other Diagrams

At this point, OVVV array can be nearly dropped. Two more diagrams are needed:

- $V^{ij}_{qs} t^q_a t^s_b$ - no problem
- $V^{ie}_{qs} t^j_e$ - no problem
- $V^{ie}_{sq} t^j_e$ - Houston ...
  we have exchange integral. The simplest solution and the best is to reevaluate integrals
- VT1 terms are tricky - to still use symmetry, need to compute $U(i,j,qs) += C(i,p) * t(j,r) * V(p,q,s,r)$ and $U(j,i,qs) += t(i,p) * C(j,r) * V(p,q,s,r)$

Now CC can be implemented much simpler with the cost of secondary storage $O(N^2 O^2)$ and per process memory $O(NO^2)$
MP2 gradient expressions have similar OVVV terms

Array implementation

- Interface that provides put/get operation
- Interfaces can be implemented for various storage backends: disk, distributed memory
- Disk-based storage is the fallback if not enough distributed memory is available
- Implementations can be switched at the runtime
- Provides transparent disk/memory algorithms

- In CPU version, matrix transformations accounts for 60 percent of runtime, integrals take up around 15 percent
- CUBLAS added, CUBLAS is driven in streams, meaning the GPU does work without CPU involved
- Now integrals take more time than they should? E.g. instead of 17 minutes, 45 minutes ...
  But only if running with other threads. Bus contention?

Performance

C12H10 cc-PVTz:
TOTAL NUMBER OF MOS = 494
NUMBER OF OCCUPIED MOS = 41
NUMBER OF FROZEN CORE MOS = 12
NUMBER OF FROZEN VIRTUAL MOS = 0
CC/DDI: needs *two* 24GB nodes to run: 70 mins per it, 12 cores

- secondary storage: 16GB, *one* node
- 1 cores: who knows ...
- 2 cores+GPU: 120 mins
- 6 cores: 80 mins
- 6 cores+GPU: 60 mins, should be around 25 mins, integrals suddenly take more time
- Parallel version as soon as I figure out ARMCI: ARMCI_MAX_THREADS

Some programming samples?

Aknowledgements

- Dr. M.S.Gordon
- Dr. R.Olson