Parallel RI-MP2

Matt Goldey

Møller–Plesset perturbation theory

- Common correction for energies obtained by the Hartree-Fock Self-Consistent Field method
- Computational costs are significant for large systems, involving polynomial scaling, depending on approximations used

Merit

- Inclusion of electron correlation

$$E_{\mathrm{MP2}} = \sum_{i,j,a,b} \langle \varphi_i(1)\varphi_j(2)|r_{12}^{-1}|\varphi_a(1)\varphi_b(2)\rangle \times \frac{2\langle \varphi_a(1)\varphi_b(2)|r_{12}^{-1}|\varphi_i(1)\varphi_j(2)\rangle - \langle \varphi_a(1)\varphi_b(2)|r_{12}^{-1}|\varphi_j(1)\varphi_i(2)\rangle}{\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b},$$

Images from Wikipedia
<http://en.wikipedia.org/wiki/M%C3%B8ller%E2%80%93Plesset_perturbation_theory>

- Contributions represented as (ia|jb)
- Formed via integral transformations from SCF calculations

RI approximation

Instead of directly forming (ia|jb), (ia|X) is formed for the auxiliary basis X (also denoted P,Q)

$(P|Q)^{-1/2}$ used to form $B_{ia}=(ia|P)(P|Q)^{-1/2}$

Largest costing step is multiplying $(ia|jb)=B_{ia}*B_{jb}$ for all a,b, auxiliary functions

Workload

Need to span virtual space for each occupied pair (through all occupied space)

Simplest implementation:

Loop i

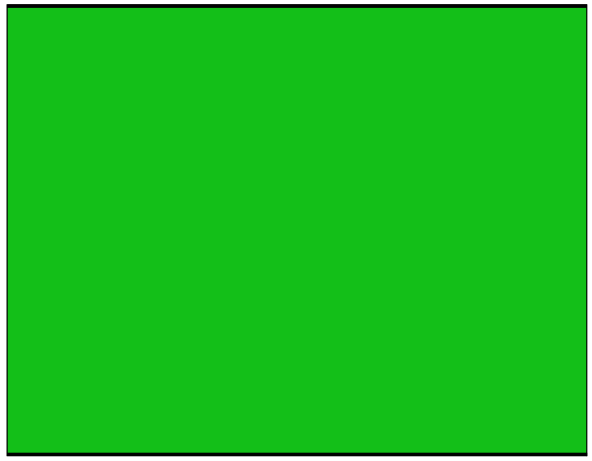
Read B_{ia} $\forall$ a,Q

Loop j

Read B_{jb} $\forall$ b,Q

Form $(ia|jb)=B_{ia}*(B_{jb})^T$

Calculate Energy contributions

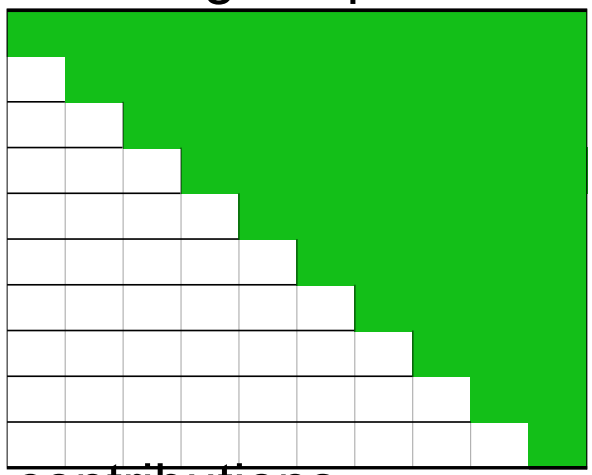


$N_{occ}*N_{occ}$ matrix multiplications performed

$N_{occ} + N_{occ}*N_{occ}$ B matrices read from hard drive -- ~$N_{occ}^2*N_{virt}*X$ in hard drive reads

Algorithm Optimization

Current description of serial algorithm not complete -- only need to span unique pairs of i,j --the upper or lower triangular portion of the occupied space



Loop i

Read B_{ia} $\forall$ a,Q

Loop j≥i

Read B_{jb} $\forall$ b,Q

Form $(ia|jb)=B_{ia}*(B_{jb})^T$

Calculate weighted energy contributions

$N_{occ}*(N_{occ}+1)/2$ matrix multiplications performed -- computation cut by 1/2

$N_{occ} + N_{occ}*(N_{occ}+1)/2$ B matrices read from hard drive -- $\sim N_{occ}(N_{occ}+1)/2*N_{virt}*X$ in hard drive reads

Even this is not optimized since reading in B_{jb} is unnecessary for j==i

Further, given that these matrix multiplications are relatively small, want to distribute work among processors

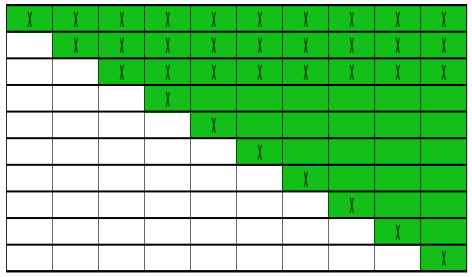
Memory Considerations

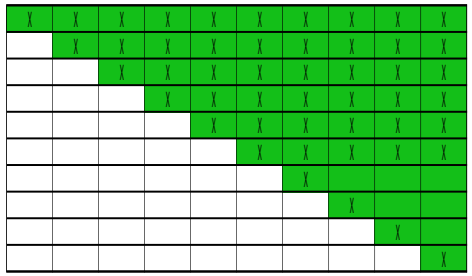
- B_{ia} & B_{jb} are (N_{virt} * X) long and are read in from hard drive as needed
- Serially, $2*(N_{virt}*X)+N_{virt}*N_{virt}$ stored in memory
- Memory delimited by earlier step to $3*X^2$
- Given N_{virt}~N, X~4*N,
  - Memory considerations for largest possible systems are $3*4*4*N^2=48*N^2$
  - Memory occupied using this serial implementation for n processor shared memory system is $n_{proc}*(2*(N_{virt}*X)+N_{virt}*N_{virt})$~$n_{proc}*(9*n^2)$
  - For 8 processors, this amounts to $72*N^2$ in memory at a time, limiting system size unnecessarily -- need a different algorithm for parallel computation for a work-distributed system

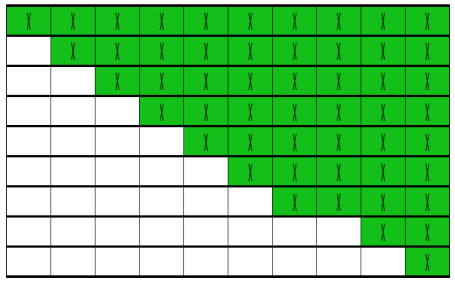
Memory-guided Work Distribution

- Need to minimize memory usage as well as minimize number of reads from the hard drive -- which is expected to become dominant in the large system limit
- Have to satisfy both criteria in work distribution scheme

Memory-guided Work Distribution







Solutions:

- Hard drive read batching -- using the space available
- Pivoting -- using what is in memory to direct the next batch

Loop over batches (determined by memory)

- Read B_{ia} $\forall$ i $\in$(first batch-1 elements of batch)
- Loop over $N_{occ} \notin$batch (N_{occ}>batch)
  - Form all possible (ia|jb), including i==j (ia|ib)

This provides a solution to the serial bottleneck of hard drive reads

Reduces the number of matrices read from ~N_{occ}^2 (initial implementation)

- New hard drive I/O costs:
- $N_{reads}=N_{occ}+(N_{occ}-(batchsize-1)-1)+(N_{occ}-2*(batchsize-1)-1)+\ldots$
- $N_{reads}=N_{occ}(N_{batches}+1)-N_{batches}-(batchsize-1)(N_{batches}(N_{batches}+1)/2)$
- N_{reads}~Nocc*$N_{batches}-(batchsize-1)N_{batches}^2/2$
- N_{reads}~$N_{occ}^2/(batchsize-1)-N_{occ}^2/(2*(batchsize-1))$
- ~=$N_{occ}^2/(2*(batchsize-1))$

This solves the hard drive read problem in serial, but does it work for parallel?

Batch size limitations

Typical system sizes constrain the batch size to a maximum of ~9-11 B matrices in memory at once

Using 9 B matrices, ignoring edge cases, 8 computations must be performed at each step, with each computation of equivalent cost

However, this method requires synchronization after each matrix multiplication -- a potential bottleneck for anisotropic systems

Distributed Memory, the Future

Further modeling is needed for extending this method to distributed memory systems since batches will not have equivalent workloads -- the occupied space needing to be spanned is diminished by each step