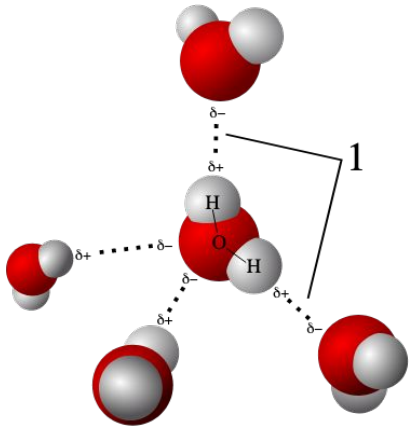
Parallelized MP2 calculation

Team 10
Hieu Dinh, Artur Lyssenko, Mouza Almualla, Olga Borodina

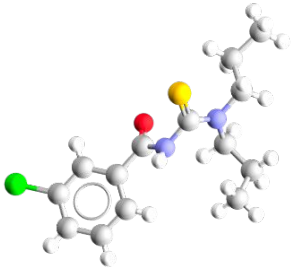
Quantum Chemistry

- Hartree Fock is the basis for all quantum chemistry calculations (99% of energy)
- Misses 1% arising from electron correlation
- Second order Möller-Plesset perturbation theory improves on
- This allows subtle behavior such as hydrogen bonding to be observed

$$E_{\mathrm{RMP2}} = -\sum_{ij}\sum_{ab}\frac{(ia|jb)[2(ia|jb)-(ib|ja)]}{\varepsilon_a+\varepsilon_b-\varepsilon_i-\varepsilon_j}$$



MP2 pipeline



Database (CCDC) or IQmol → Molecular coordinates, Basis specification → QChem (HF) → B matrix, Orbital energy (∈) → Implement + Parallelize → MP2 correction

MO_energy	(NBASIS)
C	(NBASIS, NBASIS)
$(uv \mid P)$	(NAUX, NBASIS · NBASIS)
$(P \mid Q)^{-1/2}$	(NAUX, NAUX)

```
$molecule
0 1
C        -2.29033        -2.42452         1.05258
C        -1.58865        -3.63367         1.12534
C        -0.22870        -3.67247         0.79970
C         0.42930        -2.50534         0.40201
C        -0.27271        -1.29482         0.32906
C        -1.63054        -1.25539         0.65405
H        -3.34222        -2.39417         1.30439
H        -2.09385        -4.54041         1.43283
H         0.31160        -4.60944         0.85692
H         1.48138        -2.53723         0.15047
H         0.23066        -0.38746         0.02189
H        -2.16647        -0.31616         0.59530
$end

$rem
jobtype = sp
method = hf
basis = cc-pvtz
aux_basis = rimp2-cc-pvtz
scf_guess = autosad
$end
```

```
2
6
36
-7.8309055851
-2.3612
-0.2501
0.0733
0.1621
0.1621
0.4326
```

MO_energy of LiH

MP2 algorithm

- Step 1: Change of basis

$O(\mathrm{NBASIS}^3 \times \mathrm{NAUX})$

$$(pq|P) = \sum_{q}^{N+M} \left(C^{\dagger}\right)_{q\nu} \sum_{p}^{N+M} \left(C^{\dagger}\right)_{p\mu} (\mu\nu|P)$$

for $p < N+M, P < NAUX, \nu < N+M$ **do**
  for $\mu < NBASIS$ **do**
    $(p\nu|P) \leftarrow (C^{\dagger})_{p\mu}(\mu\nu|P) + (p\nu|P)$;
  end
end
for $p < N+M, P < NAUX, q < N+M$ **do**
  for $\mu < NBASIS$ **do**
    $(pq|P) \leftarrow (C^{\dagger})_{q\nu}(p\nu|P) + (pq|P)$;
  end
end

- Step 2: Obtain B_matrix (Orthogonalization)

$O(\mathrm{NBASIS}^2 \times \mathrm{NAUX}^2)$

$$B_{pq}^{P} = \sum_{P}^{N+M} (pq|P)\,(P|Q)^{-1/2}$$

$B_{pq}^{P} = 0$;
for $p, q < N+M, P < NAUX$ **do**
  for $Q < NAUX$ **do**
    $B_{pq}^{P} = B_{pq}^{P} + (pq|Q)\,(Q|P)^{-1/2}$;
  end
end

- Step 3: Calculate MP2

$O(\mathrm{NELECTRONS}^2 \cdot \mathrm{NBASIS}^2 \cdot \mathrm{NAUX})$

$$E_{MP2} = \sum_{ij}\sum_{ab} \frac{(ia \mid jb)[2(ia \mid jb) - (ib \mid ja)]}{\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b}$$

$$(ia|jb) = \sum_{P} B_{ia}^{P} B_{jb}^{P}$$

for $i, j < N$ **do**
  for $N \leq a, b < N+M$ **do**
    $\Delta_{ij}^{ab} = \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b$;
    for $P < NAUX$ **do**
      $(ia|jb) \leftarrow (ia|jb) + B_{ia}^{P} B_{jb}^{P}$;
      $(ib|ja) \leftarrow (ib|ja) + B_{ib}^{P} B_{ja}^{P}$;
    end
    $E_{MP2} = E_{MP2} + \frac{1}{\Delta_{ij}^{ab}} \times [(ia|jb)\,[2\,(ia|jb) - (ib|ja)]]$;
  end
end

Why do we need to parallelize?

Lots of calculation :

Algorithm 1: Serial Implementation

Data: Orbital energy(ϵ), $(\mu\nu|P)$ matrix, $(P|Q)^{-1}$ matrix,
Result: MP2 correlation energy
for $i, j < N$ **do**
  for $N \leq a, b < N{+}M$ **do**
    $\Delta_{ij}^{ab} = \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b$;
    for $P < NAUX$ **do**
      $(ia|jb) \leftarrow (ia|jb) + B_{ia}^P B_{jb}^P$;
      $(ib|ja) \leftarrow (ib|ja) + B_{ib}^P B_{ja}^P$;
    end
    $E_{MP2} = E_{MP2} + \frac{1}{\Delta_{ij}^{ab}} \times [(ia|jb)\,[2\,(ia|jb) - (ib|ja)]]$;
  end
end

$O(\mathrm{NELECTRONS}^2 \cdot \mathrm{NBASIS}^2 \cdot \mathrm{NAUX})$

→ OpenMP

Why do we need to parallelize?

Lots of calculation :

Algorithm 1: Serial Implementation

Data: Orbital energy(ϵ), $(\mu\nu|P)$ matrix, $(P|Q)^{-1}$ matrix,
Result: MP2 correlation energy
for $i, j < N$ **do**
  for $N \leq a, b < N{+}M$ **do**
    $\Delta_{ij}^{ab} = \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b$;
    for $P < NAUX$ **do**
      $(ia|jb) \leftarrow (ia|jb) + B_{ia}^{P} B_{jb}^{P}$;
      $(ib|ja) \leftarrow (ib|ja) + B_{ib}^{P} B_{ja}^{P}$;
    end
    $E_{MP2} = E_{MP2} + \frac{1}{\Delta_{ij}^{ab}} \times [(ia|jb)\,[2\,(ia|jb) - (ib|ja)]]$;
  end
end

→ OpenMP

$O(\text{NELECTRONS}^2 \cdot \text{NBASIS}^2 \cdot \text{NAUX})$

Lots of data:

Molecule	NBASIS	NAUX	NELECTRONS	Memory
NH_3	72	171	5	7MB
C_6H_6	264	666	21	350MB
C_{60}	1800	4860	180	110GB

→ MPI

$\text{NBASIS}^2 \cdot \text{NAUX}$

Algorithm: MPI implementation

Data: Orbital energy(ϵ), B_{ia}^{P} matrix
Result: MP2 correlation energy

```
MPI_Init_thread(&argc, &argv, MPI_THREAD_MULTIPLE, &provided);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

#pragma omp parallel for collapse(4) schedule(static, CHUNKSIZE)
for $i, j < N$ **do**
  for $N \leq a, b < N+M$ **do**
    $E_{MP2} \leftarrow 0$;
    $\Delta_{ij}^{ab} = \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b$;
    for $P < NAUX$ **do**
      $(ia|jb) \leftarrow (ia|jb) + B_{ia}^{P} B_{jb}^{P}$;
      $(ib|ja) \leftarrow (ib|ja) + B_{ib}^{P} B_{ja}^{P}$;
    end
    $E_{MP2} = E_{MP2} + \frac{1}{\Delta_{ij}^{ab}} \times [(ia|jb)\,[2\,(ia|jb) - (ib|ja)]]$;
  end
end

```
MPI_Reduce(&E_MP2,final, &E_MP2, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
MPI_Finalize();
```

Algorithm: OpenMP implementation

Data: Orbital energy(ϵ), B_{ia}^{P} matrix
Result: MP2 correlation energy
MPI_Init_thread(&argc, &argv, MPI_THREAD_MULTIPLE, &provided);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
#pragma omp parallel for collapse(4) schedule(static, CHUNKSIZE)
for $i, j < N$ **do**
  for $N \leq a, b < N+M$ **do**
    $E_{MP2} \leftarrow 0$;
    $\Delta_{ij}^{ab} = \epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b$;
    for $P < NAUX$ **do**
      $(ia|jb) \leftarrow (ia|jb) + B_{ia}^{P} B_{jb}^{P}$;
      $(ib|ja) \leftarrow (ib|ja) + B_{ib}^{P} B_{ja}^{P}$;
    end
    $E_{MP2} = E_{MP2} + \frac{1}{\Delta_{ij}^{ab}} \times [(ia|jb)\,[2\,(ia|jb) - (ib|ja)]]$;
  end
end
MPI_Reduce(&$E_{\mathrm{MP2,final}}$, &E_{MP2}, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
MPI_Finalize();

Things to address

- What problem are you trying to solve with your application? **(Artur)**
- Describe the mathematical model and/or the data for your application in detail. Example questions that could be answered: **(Hieu)**
  a. What are the limitations of the model?
  b. In what context do you apply the model?
  c. Where does the data come from?
  d. How is the data transformed?
- Justify the effort required to parallelize the application (memory and storage requirement). Why do you need a parallel implementation of this application? **(Olga)**
- Provide a draft of how you plan to parallelize the application. What are the work packages do you plan to follow? Will you be targeting a shared memory model, distributed memory model or both? Explain the "why" here as well. **(Mouza)**

Next milestone (Parallel Design)

Your team needs to present the design of your parallel application that covers the following sections:

- You should have a sequential baseline of your application at this point for which you can present results for a simple test case. Results may be presented using visuals such as simple graphs, contour plots, volume renderings or movies for example.
- Present a profiling of your sequential baseline to identify the bottlenecks and present a simple roofline analysis of the identified compute kernels.
- Based on your analysis above propose the forms of parallelism you want to exploit in your application and which parallel programming models you will use. (This may deviate from the draft you presented in the previous presentation.)
- Elaborate on how you plan to implement the parallel code in terms of logic. What is the sequence of computational steps? Where are synchronization points? Comment on the communication overhead you expect and whether you expect load imbalance issues. Propose methods to hide these latencies.