Scaling and performance of quantum Monte Carlo

Pablo López Ríos

QMC school at S.N.Bose Centre

27 March 2015

Scaling of QMC

Usual statements about QMC:

- QMC scales with system size as N^3
- QMC scales with atomic number as $Z^{4.5}$
- QMC is perfectly parallel

How do these come about?
How do they hold up in practice?
How can one improve them?

Time per QMC step (N single-electron moves)

Typical plane-wave calculation:

- Compute N times N orbitals: $\mathscr{O}(N^3)$
- Compute N determinant ratios: $\mathscr{O}(N^2)$
- Compute N Jastrow factors: $\mathscr{O}(N^2)$
- Update inverse of cofactor matrix N times: $\varepsilon\mathscr{O}(N^3)$
- Compute local energy: $\mathscr{O}(N^2)$

For $N \ll 1000$, leading contribution to cost per step is $\mathscr{O}(N^3)$

Time per QMC step (N single-electron moves)

In a typical blip/Gaussian calculation:

- Compute N times N orbitals: $\mathcal{O}(N^2)$
- Compute N determinant ratios: $\mathcal{O}(N^2)$
- Compute N Jastrow factors: $\mathcal{O}(N^2)$
- Update inverse of cofactor matrix N times: $\varepsilon\mathcal{O}(N^3)$
- Compute local energy: $\mathcal{O}(N^2)$

For $N \ll 1000$, leading contribution to cost per step is $\mathcal{O}(N^2)$

Time per QMC step (N single-electron moves)

Localized, sparse basis (α orbitals $\neq 0$ per electron):

- Compute N times N orbitals: $\mathcal{O}(\alpha N)$
- Compute N determinant ratios: $\mathcal{O}(\alpha N)$
- Compute N Jastrow factors: $\mathcal{O}(N^2)$
- Update inverse of cofactor matrix N times: $\varepsilon\mathcal{O}(\alpha N^2)$
- Compute local energy: $\mathcal{O}(N^2)$

For $N \ll 1000$, leading contribution to cost per step is $\mathcal{O}(N^2)$

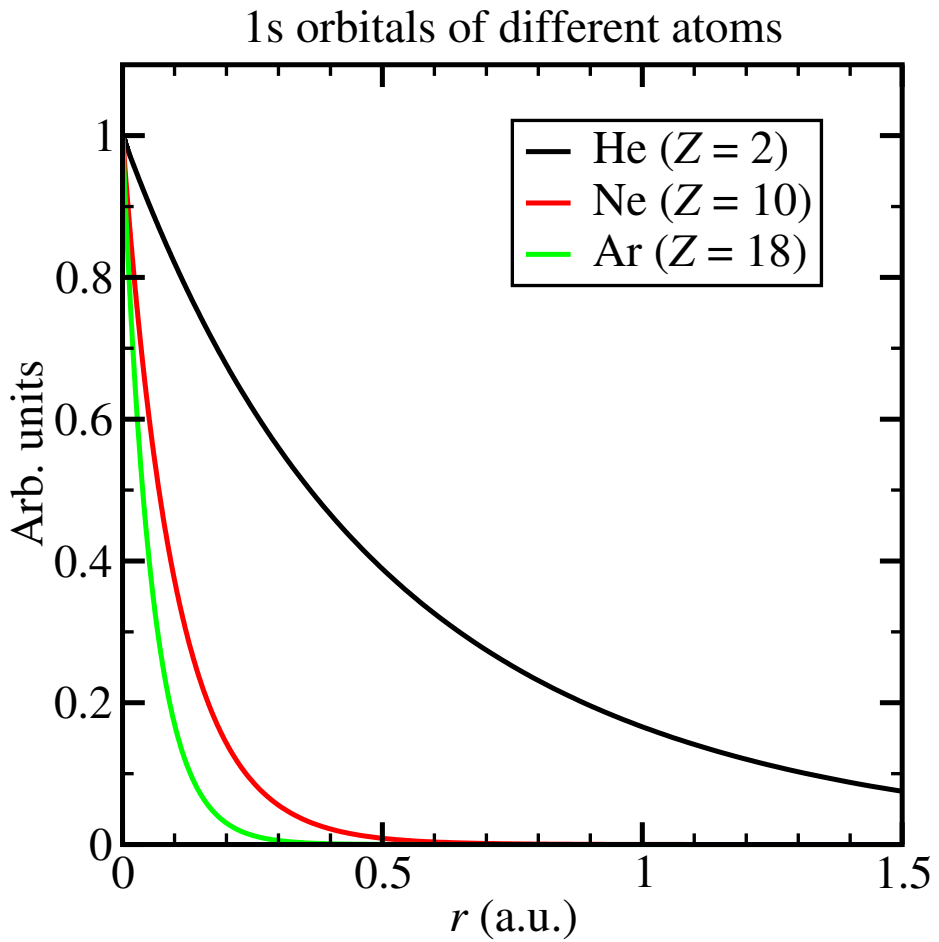
Run length requirements

How does the number of steps M scale with N?

- For fixed error in total energy: $M \propto N$
- For fixed error in energy per atom / electron: $M \propto 1/N$

Leading order in overall cost can vary from $\mathscr{O}(N)$ to $\mathscr{O}(N^3)$

Why $Z^{4.5}$ and not Z^3?



Increasing Z of all-electron atom reduces length scale of wave function

Why $Z^{4.5}$ and not Z^3?

- Small length scale require sampling with small timesteps
- Small timesteps imply increased serial correlation
- Theoretical estimates hint at $Z^{5.5-6.5}$, but tests scale as $Z^{4.5}$
- Still, heavy atoms are intractable as all-electron
- Must use pseudopotentials for heavy atoms

Memory used by blips

- Memory required by blips can become very large (several GiB)
- Current computer architectures have c *CPU cores* per *computer node*, and these cores share a single pool of RAM
- By default we run c independent processes per computer node $\rightarrow$ we store the blip coefficients c times in a node

Memory used by blips

- In case of memory issues with blips we have three options:
  - Share blip storage within a node:
    - Compile CASINO with `shm` feature enabled
    - Run with `runqmc --shm`

    You should do this in general for large blip calculations
  - Can also use openMP (multithreading) to reduce number of processes per node (this leaves some cores partially idle):
    - Compile CASINO with `openmp` feature enabled
    - Run with `runqmc --tpp=`*threads-per-process*

    In general, don't use this
  - Can also simply use fewer cores per node (this leaves some cores completely idle):
    - Run with `runqmc --ppn=`*processes-per-node*

    Do this if shared memory does not work for technical reasons

Parallel scaling of VMC

- In an M-step VMC run on C processes we run C independent random walks of length M/C
  $\rightarrow$ total cost $\propto C \times M/C = M$
  $\rightarrow$ total cost independent of C
- M forced to be multiple of C, prevents $C > M$
- M_e-step equilibration must be run in full on each process
  $\rightarrow$ cost of equilibration $\propto M_e C$
  $\rightarrow$ problematic for very large C ($C \gtrsim 10{,}000$)

Parallel scaling of optimization

- In an M-configuration optimization on C processes we independently process M/C configurations on each process
  $\rightarrow$ total cost $\propto C \times M/C = M$
  $\rightarrow$ total cost independent of C

- M forced to be multiple of C, prevents $C > M$

Parallel scaling of DMC

- In an M-step DMC run with a target population of P on C processes we evolve $p_i(t)$ configurations on the i-th process, where $\sum_i p_i(t) = P(t) \approx P$
- $p_i(t+T) \neq p_i(t)$ and $p_i(t+T) \neq p_j(t+T)$ in general $\rightarrow$ need **load-balancing** (send configurations between processes)
- Many processes can be used by increasing $P \propto C$ and reducing $M \propto 1/C$, giving same uncertainty and total cost, but:
  - Frequent checkpointing becomes problematic: set `checkpoint: 0`
  - When $P \sim C$ load balancing issues reduce the parallel efficiency: could consider using weighted DMC with `lwdmc_fixpop: T` (but avoid this)
  - M_e-step equilibration cannot be shortened as C increases

Summary

Recommendations:

- Use an **efficient basis set** (blips instead of plane waves)
- Whenever possible use **localized** basis sets
- Use **pseudopotentials** for heavy-atom (or many-atom) calculations
- **Limit parallelization** to tens of thousands of processes at most