International Journal of Computer Trends and Technology
ISSN: 2231 – 2803 / https://doi.org/10.14445/22312803/IJCTT-V69I2P109

Volume 69 Issue 2, 61-63, February 2021

Original Article

Modeling The Runtime of Guassian In Quantum Computing Methods

S. Kayathri

Department of Physics with Computer Applications, Vidhya Sagar Women's College, Chengalpattu District – 603 111, Tamil Nadu, India.

Received Date: 29 December 2020
Revised Date: 10 February 2021
Accepted Date: 13 February 2021

Abstract - *In quantum computing methods, density functional theory (DFT) is the most powerful approach to calculate the electronic structure of physical systems containing a large number of atoms. Currently, a variety of computational methods that implement DFT equations in the basis set of plane waves, Gaussians, localized numerical orbital using real-space representation. There is a huge interest to make further progress in the modeling development of electronic structure calculations. In that regard, so many different and complementary research directions are currently pursued worldwide. One direction is devoted to developing methods that give accurate results in cases where standard approximations in DFT these developments include much fundamental theory.*

Keywords - *B3LYP, cache line, DFT, Guassian, Hartree-Fock, optimization.*

I. INTRODUCTION

This paper explores the performance of the computational model, which determines execution time-based instruction. It misses counts of the cache using the behavior of the two-electron integral algorithm in the Gaussian computational package. Four different computational platforms are considered with a total of seven individual microprocessors. Hartree-Fock and hybrid Hartree-Fock of Density Functional Theory electronic structure methods are assessed. In most cases, the model is found to be accurate less than 3%.

II. B3LYP

The following outlines the procedure for the computation of GEO type orbitals and for the prediction of α and Hatree Fock for the prediction of β. All calculations for these examples should be run only on a standard laptop computer. The calculations have been carried out using Gaussian and the associated visualization package Gauss view.

A. Gauss view

Gaussian and Gauss view is a short introduction to building a molecule and running an optimization. It is used to calculate the molecular structure, and the Gauss view helps to implement the "Basic Molecule Building" tutorials.

The running time of different types of computational modeling methods depends on molecular mechanics and ab initio calculations, software available for computational calculations, and wave function. The wavefunction (ψ) is dependant on all elections. It can be represented by the product of individual electron wave functions for a single orbital.

$$\psi(r1...rn) = \varphi1(r1)\varphi2(r2)...\varphi n\ (rn)$$

To solve this Schrödinger equation, the repulsion of the electron-electron effect must be substituted with Hartree's theory. The effective repulsion is represented by Vi eff. The average positions of remaining electrons replaced this exact repulsion is with an effective field which is a major assumption of computation. The total electronic energy can be described as the sum of electronic energies Ei with the assumption of one electron wavefunction. It arises from the Hartree equation:

$$(K + Ve\text{-}n + Vi\ eff)\ \varphi i = Ei\ \varphi i$$

– K is the kinetic energy operator for electrons
– V is the potential energy operator due to electron nucleus attraction
– Vi eff is the effective field of all electrons

B. Linear Combination Of Atomic Orbitals (LCAO)

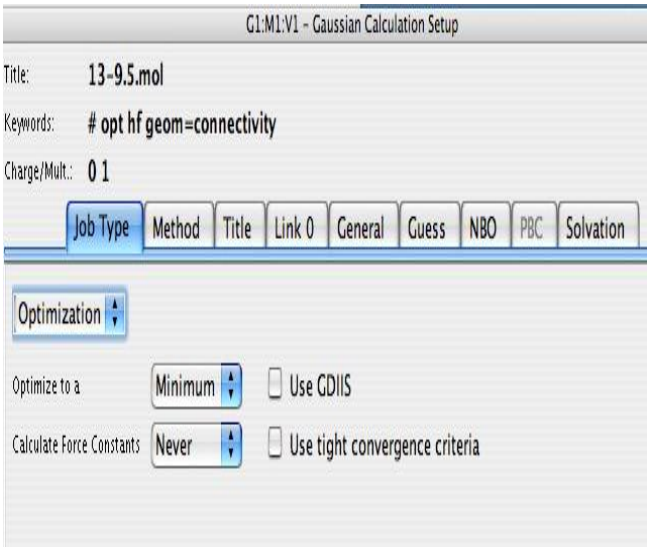
Electronic wavefunctions φi are molecular orbitals that span the entire molecule. As a further simplifying approximation, molecular orbitals are constructed as the sum of atomic orbitals χu Because HF theory uses an effective electron-electron repulsion term, HF energy, EHF will always be greater than the exact energy E. The electron-electron repulsion is referred to as electron correlation:

$$Ecorr = E - EHF$$

This is the best-case error of HF theory. Gaussian Website Exchange of basis sets and pseudo-potentials and some basic knowledge of running calculations of optimization.

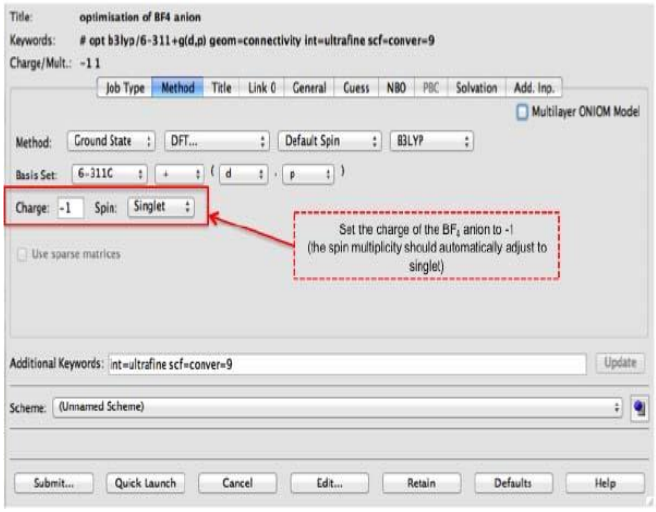
III. OPTIMIZATION

Properties must be computed before the selected cations and anions must first be optimized. It is confirmed as minima on their respective potential energy surfaces via frequency analysis. While these can be run as a single job that optimizes and then runs the frequency analysis.



The optimization does not work the first time or is not complete. It takes a long time and wastes computer resources. All calculations here have been performed using B3LYP/6-311+g(d,p) with the additional keywords int-ultrafine and scf-conver-9. To set calculations with these options, improve the criteria convergence and integration grid over the Gaussian defaults. To work with ions, it is important to remember to adjust the charge of the Guassian system. This image is a setup-up like in gauss view.

At the B3LYP/6-311+g(d,p) level, normally expect the six lowest frequencies to be within, or close to, +/- 10cm^{-1}. From the optimized structure, it is easy to confirm and calculate the properties of the molecule



IV. MODELING CACHE INTERFERENCE

All data to be reused map to different cache locations, then the number of cache misses per variable is simply D(v)/ R(v), where D(v) is the total number of memory references of the variable v and R(v) is the factor of reuse variable v. These are the intrinsic misses. Generally, we also have interference misses; if the reuse of the variable v misses at a rate of M(v), then the total number of misses for v is,

$$Mp(v) = 1 – J'J (1 – Ip(u, v)) ,Uev$$

where 1P(u, v) is the probability that accesses to the overall cache miss rate is a combination of three kinds of misses: intrinsic misses, self-interference misses, and cross-interference misses. The figure shows a breakdown of the misses into the three categories for blocked matrix multiplication on a cache of l000 words.

V. THE EFFECT OF CACHE BLOCKING ON THE B3LYP ALGORITHM

The objective of this section is to investigate the effects of cache blocking on the batch sizes and execution time for different quartet types within the algorithm. Evaluation of Hartree Fock method based on the modification of previous schemes with four-center integrals. The code is obtained in the case when integrals are calculated directly using analytical formulas.

Performance tests of the serial version of the code indicate that systems with more than 100 atoms can be calculated in less than an hour and to calculate the system with as much as 1500 atoms on a single processor with a computational time of several hours. The ability to treat systems with more than 1000 atoms using a serial version on a single CPU core implies that parallelization should enable calculations of systems with 10,000 atoms on modest size computing clusters with several hundreds of CPU cores. One HF SCF cycle on the AMD848 using a single CPU for B3LYP /6-31G* takes 13 sec., /6-31++G(3df,3pd) takes 21.4 min., 6-31G* takes 20 min. and 6-31++G(3df,3pd) takes 3.4 days.

VI. FUTURE SCOPE

A linear performance model was proposed to measure execution time for the Slater type algorithm. The α PPC eff refers to how well the code uses the superscalar resources of the processor, γ corresponds to the average cost in cycles of an L2 miss, which required cache lines to be fetched from D-RAM. It was shown that the LPM can produce good fits for measured cycle counts obtained using hardware performance counters. Using the test of molecular systems, it was found that the optimal blocking factor is both platform and computation specific. Evaluation of the LPM in conjunction with functional cache simulation shows it is able to reproduce the trends and cycle counts as a function of varying the cache blocking factor.

Data could be gathered for the first couple of SCF cycles, and the blocking factor could be varied by starting from the default measurements for an increased and decreased vales. Once the blocking factor with the lowest cycle count is measured, it can be used for the remaining SCF cycles. We also intend, for future work, to study the performance impact of shared last-level of cache on multi-core systems using the LPM.

The simple linear combination of instructions issued and cache misses,

$$\text{Cycles} = \alpha * (\text{I Count}) + \beta * (\text{L1Misses}) + \gamma * (\text{L2Misses}) \quad (1)$$

where I Count is the instruction count, L1 Misses the total number of Level 1 cache misses, L2Misses the total number of Level 2 cache misses, and α,β, and γ are penalty factors.

Increased cache line size aids scientific application of the molecule, and the design of modern microprocessors is largely driven by commercial workloads. So the sizes of cache line microprocessors are usually 64 bytes. Our cache variation experiments indicate that there is scope to reduce running time, read and write misses. This can be achieved by a series of hybrid techniques.

One possible way of achieving this in future work is through the use of a pre fetch queue which eliminates large misses. It is a Guassian queue that holds n address that needs to be pre-fetched. The queue is an effective way to handle memory references that are interspersed throughout memory. The depth of the queue, which is determined experimentally, is deep enough to ensure that values are popped off the stack.

VII. CONCLUSION

In conclusion, here presented the Gaussian basis implementation of the charge method for electronic structure calculations. These are not present in Gaussian implementations of DFT. The most convenient way to store the charge density motifs is to use a uniform real-space grid. Consequently, the method to obtain the Gaussian basis representation of the electronic charge density from quantum computing is represented in real space.

This paper shows that blocking is effective in reducing the memory access latency for caches using the combination of theory and experimentation calculation. When blocking numerical codes for a cache is used to block optimization. First, use block copying and provides the fewest cache misses. The most robust performance of the options that we have considered. If copying is not appropriate to calculate, then choose the largest block size within the array, which is used to reduce the run time.

REFERENCES

[1] N. Nethercot, A. Mycroft, The Cache Behaviour of Large Lazy Functional Programs on Stock Hardware, Proceedings New York, USA, (2002).
[2] G. H. Golub and C. F. Van Loan. Matrix Computations. Johns Hopkins University Press, (1989).
[3] M. E. Wolf and M. S. Lam. A data locality algorithm. Submitted for publication., (1990) optimizing.
[4] D. Gannon and W. Jalby., The memory of hierarchy on algorithm organization.
[5] J. Dongarra, J. Du Croz, S. Hammarling, and I. Duff. Level (III) basic linear algebra programs
[6] ACM Tanwdion.R. on Mathematical Software, (1990) 1-19.
[7] In Proceedings of Annual ACM Symposium on Theory of Computing, (1981) 326-333. ACM SIGACT.
[8] The impact of hierarchical memory systems on linear algebra algorithm design.
[9] Technical Report UIUCSRD 625, University of Illinois, (1987).