Modeling the adsorption process on graphic processors

Dmitry Vinokursky[1], Natalia Kononova[1], Yana Agakhanova[2], Dmitry Miroshnichenko[3] and Nikita Shelestov[1]

[1] Federal State Autonomous Educational Institution of Higher Education North-Caucasus Federal University, Stavropol, 355009, Russia
[2] Moscow Institute of Physics and Technology, Moscow, 117303, Russia
[3] CRT-GROUP, Moscow, 117587, Russia
knv_fm@mail.ru

Abstract. This paper briefly describes the technology of non-specialized computing on GPUs and the theory of density functional. The Quantum ESPRESSO software package was used to simulate the adsorption of an aluminum atom by the C60 fullerene. It is shown that the binding energy of aluminum and fullerene atoms depends on the aluminum atom position. The density of states is calculated for each case.

Keywords: Computing on graphics processing units, Density functional theory, Adsorption.

1 Introduction

General Purpose GPU Computing (GPGPU) has exploded in the last few years. Transferring calculations from a central processing unit (GPU) to a video chip (GPU) can often significantly speed up the computing speed. For example, according to Nvidia, GPU molecular dynamics simulations using CUDA technology provide a 24x improvement in performance over the same calculations on a CPU. The use of GPGPU technology in quantum chemical calculations made it possible to carry out calculations that previously took days in a few hours.

2 Materials and methods

2.1 GPGPU Technology

GPGPU (General-purpose graphics processing units) is a technology that uses a graphics processing unit (GPU), which typically processes computations only for computer graphics, to perform computations that are traditionally performed by a central processing unit (CPU) [1].

Originally developed in 2003 to improve shader programs, the GPGPUs have shown to be well suited to scientific computing needs, and development has continued in this direction since then.

GPGPU is a form of parallel processing of data between one or more GPUs and CPUs that parse the data as if it were in the form of an image or some other graphical form. Although GPUs run at lower frequencies, they usually have multiple cores and a fundamentally different architecture (Figure 1). This allows GPUs to process much more images and graphics data per second than CPUs. Transferring data in graphical form, scanning it and analyzing it with a GPU can significantly speed up calculations.

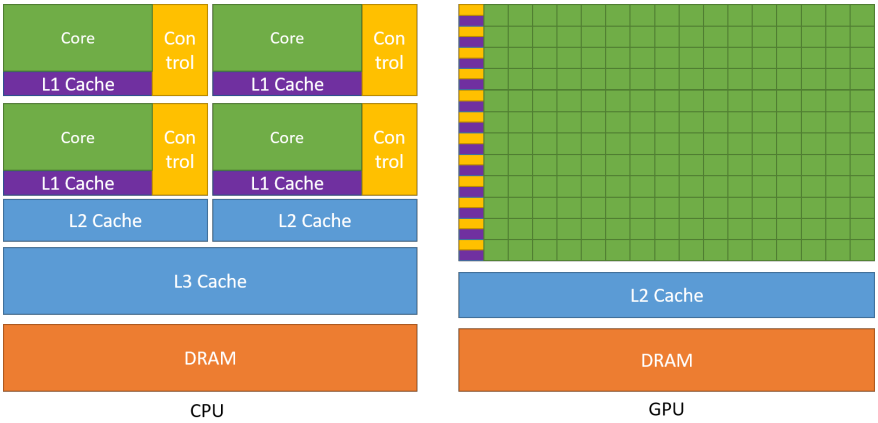

Fig. 1. GPU and CPU device. CPU is on the left, GPU is on the right.

GPUs implement the Single instruction, multiple data method (SIMD). This means that the GPU cores execute the same set of instructions for each data instance. Most graphics algorithms use this approach as the most efficient method for graphics rendering. The module that converts a stream of input data into an output is called the kernel.

The fundamental difference between CPU and GPU architectures is that if the CPU is designed to execute one stream of instructions sequentially, then the GPU assumes the simultaneous execution of a large number of parallel threads [2].

CPU performance is determined by the number of tasks executed per clock cycle. However, it is impossible to continue to increase the speed of execution of algorithms at the expense of a proportional increase in the number of cores, while maintaining the architecture of sequential execution of the instruction flow. The number of transistors that need to be added for the corresponding performance gain will very quickly become too large.

On the other hand, GPUs were designed to process large numbers of instruction streams in parallel. Moreover, these threads are natively parallelized, and there is simply no overhead of parallelizing instructions in the GPU. Due to this organization

of computations, the graphics processor uses many execution units, which, unlike the sequential flow of instructions for the central processor, are easily loaded [3, 4].

In terms of organizing access to memory, the CPU and GPU also differ significantly. If the CPU structure is focused on algorithms with random access to memory, then the GPU reading and, accordingly, writing occurs sequentially, which means that if there was an access to a memory cell, then the data that follows will be used [5]. The problem of data access latency in the CPU is solved using the caching technology and code branch prediction. The solution to the same problem with the help of a GPU looks completely different: if one of the simultaneously running processes has suspended its execution pending access to memory, the video chip switches to another process, which already has all the data necessary for further execution. The caching mechanism is used in the GPU to reduce memory access latency. However, if in the central processor the cache occupies an impressive part of the chip, then the graphics processor allocates only 128-256 kilobytes for the needs of the cache, which, in turn, is used to increase the bandwidth.

Multi-threading in GPUs is also implemented at the hardware level. It makes no sense to increase the number of threads in the CPU, since each switching between threads inevitably leads to significant delays of several hundred cycles, so the CPU core can usually only execute several threads at the same time (8-12). The GPU, in turn, is able to instantly switch between threads, and each core can handle over a thousand threads [6,7].

Video card manufacturers saw a big promise in GPGPU in 2003 and soon both of the largest GPU manufacturers presented their solutions for performing general computing on GPUs: Nvidia - CUDA, AMD - ATI Stream.

2.2 Using GPUs for Quantum Chemical Calculations

In this work, all calculations were performed using Quantum ESPRESSO. Quantum ESPRESSO is a software package based on the density functional theory (DFT) and the projected augmented wave method (PAW method) [8]. It is a powerful tool for energy calculations of multi electronic systems and is intended for simulation at the quantum mechanical level. It is distributed under the GNU General Public License and is completely open source [9].

What distinguishes Quantum ESPRESSO from its counterparts is the ability to perform calculations on GPUs that support CUDA technology, which can significantly speed up calculations even on personal computers [10].

3 Results

In 1964, Hohenberg and Kohn published a paper laying the foundation for density functional theory (DFT) [11]. The essence of DFT is to replace a complex and, therefore, difficult to calculate many-electron wave function, which contains 3N variables (N is the number of electrons, and each electron has 3 spatial variables), with an electron density functional, which contains only 3 variables [12]. So in the new system,

we don't have to worry about the huge number of 3N variables, instead we only deal with 3 variables that are much easier to deal with. Hohenberg and Kohn proposed their first theorem, which indicates that the ground state energy is uniquely dependent on the electron density, which means that it is a functional of the electron density. Their second theorem proved that by minimizing the energy of the system according to the electron density, one can obtain the energy of the ground state.

Despite the fact that the theorems of Hohenberg and Kohn confirm the truth of the fact that there is a one-to-one correspondence between the electron density functional and the properties of the system, they do not give any exact information what exactly these interactions are.

So the most commonly used method instead of "minimizing system energy" is the Kohn-Sham method. Kohn and Sham published the paper in 1965, just a year after the important paper by Hohenberg and Kohn, and in this paper they simplified the many-electron problem to the problem of noninteracting electrons in the effective potential. This potential includes the external potential and the effects of Coulomb interactions between electrons, for example, exchange and correlation interactions. Working with the exchange and correlation interactions is the main difficulty in Kohn and Sham density functional theory. There is still no rigorous way to solve the exchange and correlation energy. However, the simplest approximation is the local density approximation (LDA). The LDA is based on the use of a uniform electron gas model to obtain the exchange energy (the exact value can be obtained from the Thomas-Fermi model) and to obtain the correlation energy from fits to a uniform electron gas [13].

DFT has become very popular for computations in solid state physics since the 1970s. Compared to other methods dealing with quantum mechanical problems of multiple bodies, LDA gives satisfactory results with experimental data. But in the field of quantum chemistry, DFT was not accurate until the 1990s, when approximation methods were significantly improved to better simulate exchange correlation interactions [14]. DFT is currently the leading method for calculating electronic structure in many fields. However, it is still difficult to use DFT to work with highly correlated systems, band gaps in semiconductors, and highly dispersive systems.

In the theory of the electron density functional, the total energy of a crystal is determined by the following expression:

$$E_{tot} = \sum_{\sigma=\alpha,\beta} \sum_{\nu} f_{\nu} \left\langle \psi_{\nu}^{\sigma} \mid T \middle| \psi_{\nu}^{\sigma} \right\rangle + \sum_{\sigma\sigma'} \int w_{\sigma\sigma'} n_{\sigma'\sigma} + \frac{1}{2} \iint \frac{n(\mathbf{r}) \cdot n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} dV dV' + E_{xc}\{n_{\sigma\sigma'}\} \quad (1)$$

Here the first term is the kinetic energy of the electrons in the crystal; f_{ν} - population of orbital ν, σ - spin variable (α - spin up, β - spin down); ψ_{ν}^{σ} - wave function. The second term in expression (1) is the energy of the electron-nuclear interaction. In the equation $n_{\sigma\sigma'} = \sum_{\nu} f_{\nu} \psi_{\nu}^{\sigma} \psi_{\nu}^{\sigma'}$ - is an electric charge density operator. The third term in formula (1) is the energy of the electron-electron interaction. The last term in this formula describes the exchange-correlation interaction.

Otherwise, equation (1) can be represented as:

$$E_{tot} = E_{band} - \frac{1}{2}\int n(\mathbf{r})V_H dV - \int Tr(V_{xc} n(\mathbf{r}))dV + E_{xc} \quad (2)$$

Where E_{band} - zone energy, V_H - Hartree potential, V_{xc} - potential of exchange-correlation interaction, Tr - matrix trace computation.

4 Discussion

In this work, the adsorption of an aluminum atom on the C60 fullerene was studied. Due to the symmetry of the problem, only the following placement options are possible: aluminum on the carbon atom, aluminum above the carbon-carbon bond, and aluminum in the center of the cell on the fullerene surface (Figure 2).

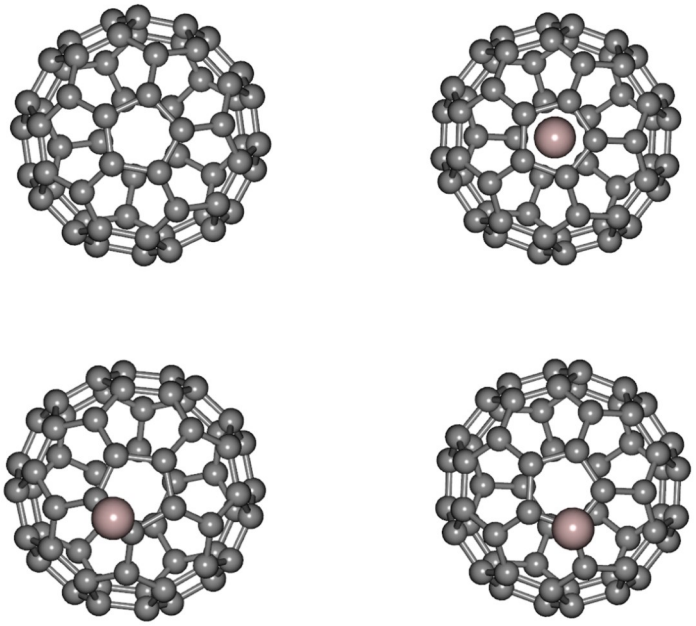

Fig. 2. The location of an aluminum atom above C60 molecule.

Figure 3 shows the density of states of pure fullerene and fullerene with an aluminum atom:

- is the density of states for fullerene C60;
- is the density of states when aluminum is positioned in the center of the cell on the fullerene surface;

– is the density of states when aluminum is positioned above the carbon-carbon bond;
– is the density of states at the position of aluminum on the carbon atom.

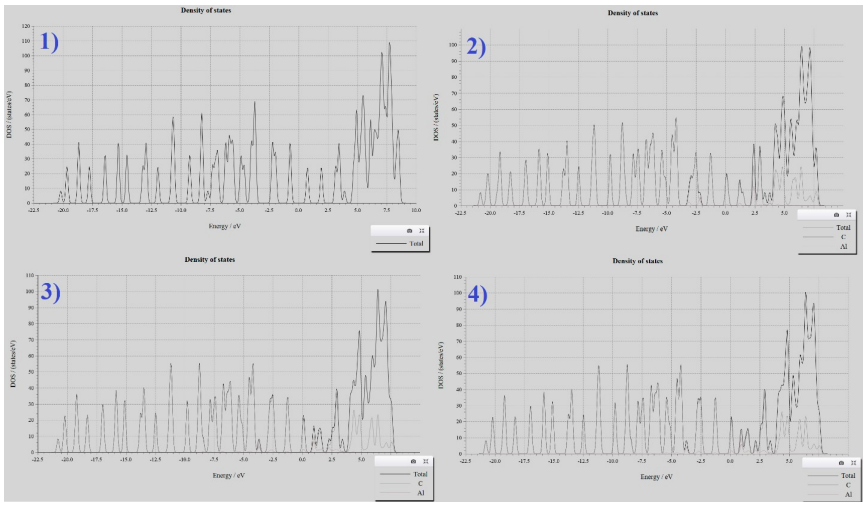

Fig. 3. Density of states.

Table 1 shows the calculated binding energies for various positions of aluminum.

Table 1. Binding energies for aluminum and fullerene atoms.

Aluminum atom over the carbon atom, eV	Aluminum atom between carbon atoms, eV	Aluminum atom in the center, eV
54.14	54.12	54.35

All calculations were performed using the Quantum ESPRESSO software package.

5 Conclusion

The calculations were performed on the GPU cluster of the North-Caucasus Federal University.

The paper shows the relationship between the binding energy and the position of the aluminum atom on the fullerene surface.

References

1. Koyano, T., Kuroiwa, Y., Kita, E., Saegusa, N., Ohshima K., Tasaki A.: J. Appl. Phys. 64, 5763 (1988).

2. Katayama, T., Yuasa, S., Velev, J.P., Zhuravlev M.Y., Jaswal S.S.: Appl. Phys. Lett., 89, 112503 (2006).
3. Faure-Vincent, J., Tiusan, C., Bellouard, C., Popova, E., Hehn, M., Montaigne, F., Schuhl, A.: Phys. Rev. Lett., 89, 107206 (2002).
4. Mathon, J., Umerski, A.: Phys. Rev. B, 63, 220403 (2001).
5. Butler, W.H., Zhang, X.-G., Schulthess, T.C., MacLaren, J.M.: Phys. Rev. B, 63, 054416 (2001).
6. Yang H.X., Chshiev M., Dieny B., Lee J.H., Manchon A., Shin K.H.: Phys. Rev. B, 84, 054401 (2011).
7. Lambert, C.-H., Rajanikanth, A., Hauet, T., Mangin, S., Fullerton E.E., Andrieu, S., Appl. Phys. Lett., 102, 122410 (2013).
8. Balogh, J., Dĭzsi, I., Fetzer, C., Korecki, J., Kozioł-Rachwał, A., Młyńczak, E., Nakanishi, A.: Phys. Rev. B, 87, 174415 (2013).
9. Kozioł-Rachwał, A., Skowroński, W., Ślęzak, T., Wilgocka-Ślęzak, D., Przewoźnik, J., Stobiecki, T., Qin, Q. H., Dijken van S., Korecki, J.: J. Appl. Phys., 114, 224307 (2013).
10. Lawler, J.F., Schad, R., Jordan, S., Kempen H. van: J. Magn. Magn. Mat., 165, 224 (1997).
11. Subagyo, A., Seuoka, K., Mukasa, K.: IEEE Trans. Magn., 35, 3037 (1999).
12. Vassent, J.L., Dynna, M., Marty, A., Gilles, B. Patrat, G.: J. Appl. Phys., 80, 5727 (1996).
13. Popova, E., Faure-Vincent, J., Tiusan, C., Bellouard, C., Fischer H., Hehn, M., Montaigne, F., Alnot, M., Andrieu, S., Schuhl, A., Snoeck, E., Costa, V. da: Appl. Phys. Lett., 81, 1035 (2002).
14. Paggel, J.J., Wei, C.M., Chou, M.Y., Luh, D.-A., Miller, T., Chiang, T.-C.: J. Appl. Phys., 97, 036104 (2005).