EPiC Series in Computing

Volume 98, 2024, Pages 1–14

Proceedings of 39th International Conference on Computers and Their Applications

Evaluating the Use of ChatGPT for Parallel Programming with OpenACC

Evaldo B. Costa[1] and Gabriel P. Silva[1]

Computer Institute, UFRJ, Rio de Janeiro, Brazil
{ebcosta,gabriel}@ic.ufrj.br

Abstract

Artificial intelligence (AI) applications are increasingly prevalent in various aspects of our daily lives. One such application is ChatGPT, which has garnered significant interest from professionals across different fields seeking to leverage AI for diverse purposes, including parallel programming. However, concerns have been raised regarding ChatGPT's ability to generate correct and efficient code.

The objective of this study is to evaluate the effectiveness of ChatGPT (GPT-3.5), an advanced language model, for parallel programming using OpenACC. Parallel programming plays a crucial role in accelerating computationally intensive applications and improving overall performance. OpenACC, a parallel programming standard, enables developers to harness the computational power of GPUs for application acceleration.

By conducting a series of experiments, we compare different approaches to parallel programming, with and without the assistance of ChatGPT. Key evaluation factors include execution performance, code quality, and user-friendliness.

The results suggest that integrating ChatGPT can positively impact the parallel programming process. However, limitations and challenges are identified, such as the need for proper parameter adjustment and reliance on training data utilized by ChatGPT. Suggestions for future research directions are provided to address the identified limitations and further enhance the integration of ChatGPT into the parallel programming workflow.

1 Introduction

Artificial intelligence (AI) has grown fast in recent years, with numerous applications in a wide range of industries. Like all other fields, education is being influenced by the development of AI-powered applications. Numerous applications for educational use have been developed, ranging from improving services and administrative processes to academic applications in various fields of knowledge [8] [6].

ChatGPT is one such application, and there have been many studies on the use of ChatGPT in education. In general, these studies have identified some issues with the quality and reliability of information generated by ChatGPT [3]. The results reveal that some of the generated data may be inaccurate; however, this does not necessarily means that it cannot be used. It is important to pay close attention to and carefully evaluate the information generated by ChatGPT [10].

Among the various possible applications for ChatGPT, automatic code generation stands out, based on initial specifications provided by the user. The final program can be obtained almost instantly compared to the time spent on code development done by human programmers. To achieve this, thousands upon thousands of lines of open-source code are analyzed, so that the AI can learn to generate code correctly, involving massive volumes of data to achieve the best potential performance [16].

Although simpler applications developed for students who are starting to practice programming can be developed with relative accuracy and correctness, we may question how well ChatGPT performs in the development of more sophisticated applications, such as parallel programming [12].

Parallel architectures, such as multicore, manycore, and GPU architectures, achieve high levels of parallelism by utilizing a growing number of processors. These designs employ massively parallel processing and are widely used in various applications. GPUs, in particular, stand out due to their emphasis on energy efficiency and high throughput demands, with NVIDIA developing the GPU in the 1990s [13].

There are some parallel programming paradigms that can be utilized in the development of programs to GPUs; one of these paradigms is OpenACC, which allows programmers to designate which parts of the code should be accelerated without requiring large changes to the sequential code.

OpenACC works with high-level compilation directives and is intended for usage in accelerators, which allow the compiler to relocate the computation to an accelerator by detecting parallel parts of the code [7] [4].

This article aims to compare the quality of automatically generated parallel programming code by ChatGPT using OpenACC to the quality of code manually generated by human programmers.

2 Related Work

The ChatGPT is a natural language processing application that employs artificial intelligence technologies to simulate interactions with real people [11]. With ChatGPT, you can ask questions about numerous topics and help in different tasks. ChatGPT was created by OpenAI and is based on OpenAI's Generative Pre-trained Transformer (GPT) language model architecture [2].

When asked the ChatGPT itself to answer what it is, we have the following response presented in Figure 1.

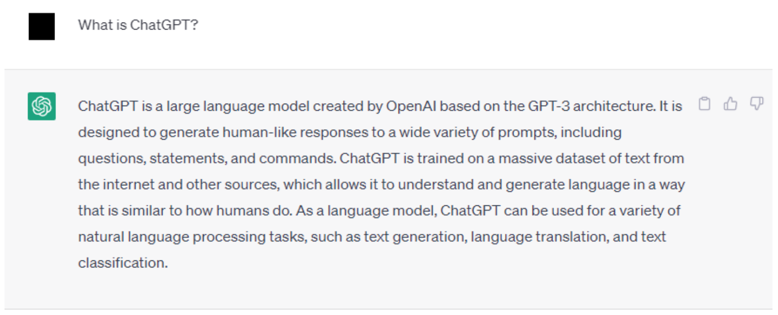
This type of generative AI model is trained on massive amounts of data. The language model was refined using both supervised and reinforcement learning [14] [15]. ChatGPT is distinguished by its use of Reinforcement Learning from Human Feedback (RLHF) [1].

This method enables ChatGPT to provide consistent, contextually relevant responses to a wide range of natural language input. Figure 2 depicts the ChatGPT training procedure.

ChatGPT is currently becoming a phenomenon, frequently utilized by many people and companies in a wide range of fields, from simple day-to-day challenge to complicated tasks that can assist in decision-making.

Students in education, like those in other fields, are already using ChatGPT to assist them in their academic work, from conducting research to carrying out exercise questions [9].

Currently, studies or research that can demonstrate the validity of texts generated by ChatGPT are lacking. Assessing the impact of using ChatGPT on science education topics or its potential applications for the creation of science is essential [5].


What is ChatGPT?

ChatGPT is a large language model created by OpenAI based on the GPT-3 architecture. It is designed to generate human-like responses to a wide variety of prompts, including questions, statements, and commands. ChatGPT is trained on a massive dataset of text from the internet and other sources, which allows it to understand and generate language in a way that is similar to how humans do. As a language model, ChatGPT can be used for a variety of natural language processing tasks, such as text generation, language translation, and text classification.

Figure 1: What is ChatGPT, according to the ChatGPT?

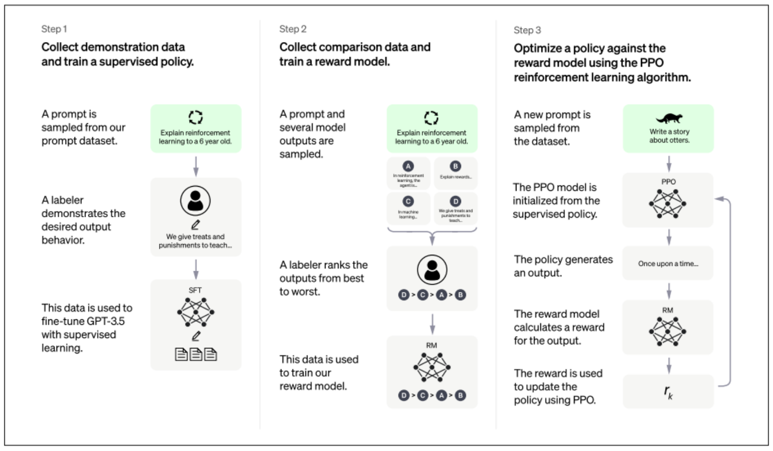

Step 1
Collect demonstration data and train a supervised policy.
A prompt is sampled from our prompt dataset.
Explain reinforcement learning to a 6 year old.
A labeler demonstrates the desired output behavior.
We give treats and punishments to teach...
This data is used to fine-tune GPT-3.5 with supervised learning.
SFT
Step 2
Collect comparison data and train a reward model.
A prompt and several model outputs are sampled.
Explain reinforcement learning to a 6 year old.
A
B
C
D
A labeler ranks the outputs from best to worst.
D > C > A > B
This data is used to train our reward model.
RM
D > C > A > B
Step 3
Optimize a policy against the reward model using the PPO reinforcement learning algorithm.
A new prompt is sampled from the dataset.
Write a story about otters.
The PPO model is initialized from the supervised policy.
PPO
The policy generates an output.
Once upon a time...
The reward model calculates a reward for the output.
RM
The reward is used to update the policy using PPO.
r_k

Figure 2: The ChatGPT training process [10]

Considering this context, we aim to compare the efficiency and effectiveness of codes generated with and without ChatGPT for parallel programming using graphical processing units (GPUs) and OpenACC in this paper.

3 Methodology

We will divide our research into two steps to begin. In the first step of the study, we asked ChatGPT (GPT-3.5) to generate three codes parallel programming in C for GPUs using OpenACC.

The following step, we will write the code for the same applications. Using strategies for obtaining the best performance in each application.

The codes utilized in this article are readily accessible through a link that can be downloaded: https://github.com/evaldocosta/chatgpt.

Codes generated from the following applications will be used for the evaluation:

- Calculate fractal Mandelbrot

Fractals are complex geometric shapes characterized by self-similarity. The Mandelbrot fractal is defined as the set of complex plane points C. Complex numbers are subjected to an iterative and recursive procedure that generates the Mandelbrot set.

- Calculate matrix multiplication

Matrix multiplication, also known as matrix product and two-matrices multiplication, yields a single matrix. It's a binary operation. If A and B are two matrices, then their product is denoted by: X = A x B.

- Calculate Jacobi interactive

The Jacobi method is an iterative procedure for solving a system of diagonally dominant linear equations. Each diagonal element is found and an approximate value is entered. The technique is then repeated until convergence is reached.

- Calculate prime number

Prime numbers are those that can only be divided by themselves and one; otherwise, dividing them by another integer yields a non-whole number. As a result, dividing the number by anything other than one or itself yields a non-zero remainder.

4 ChatGPT-created codes

4.1 Calculate fractal Mandelbrot

To create the Mandelbrot code, we asked ChatGPT the following question: "Create mandelbrot calculation C programming code for GPU processing using OpenACC.".

In Listing 1 shows the answer with a part of the code using OpenACC.

```
#pragma acc data create(buffer[0:WIDTH*HEIGHT])
#pragma acc parallel loop collapse(2) private(iter)
    for (i = 0; i < HEIGHT; i++) {
        for (j = 0; j < WIDTH; j++) {
            float x = x_min + j * dx;
            float y = y_min + i * dy;
            float zx = 0.0, zy = 0.0;
            iter = 0;
            while (iter < MAX_ITER && (zx * zx + zy * zy) < 4.0) {
                float new_zx = zx * zx - zy * zy + x;
                float new_zy = 2.0 * zx * zy + y;
                zx = new_zx;
                zy = new_zy;
                iter++;
            }
            buffer[i * WIDTH + j] = iter % 256;
        }
    }
```

Listing 1: A screenshot of the ChatGPT answer with C code from the OpenACC snippet

After ChatGPT generates the code, the system makes some notes about it. The observations written by ChatGPT are as follows:

The code above uses OpenACC directives to parallelize the Mandelbrot set computation over the GPU. The #pragma acc parallel loop directive instructs the compiler to parallelize the loop over the GPU, and the #pragma acc data create directive creates a data region that transfers the buffer array to the GPU for computation.

Note that in order to compile and run this code, you need a compiler that supports OpenACC, such as PGI, Cray, or GNU compilers. You also need to link against the OpenACC runtime library, which is usually provided by the compiler vendor.

4.2 Calculate matrix multiplication

To create the matrix multiplication code, we asked ChatGPT the following question: ”Create matrix multiplication C programming code for GPU processing using openACC.”.

In Listing 2 shows the answer with a part of the code using OpenACC.

```
#pragma acc data copyin(A[0:N*N], B[0:N*N]), create(C[0:N*N])
{
    #pragma acc kernels
    {
        #pragma acc loop independent collapse(2)
        for (i = 0; i < N; i++) {
            for (j = 0; j < N; j++) {
                float temp = 0.0f;
                for (k = 0; k < N; k++) {
                    temp += A[i * N + k] * B[k * N + j];
                }
                C[i * N + j] = temp;
            }
        }
    }
}
```

Listing 2: A screenshot of the ChatGPT answer with C code from the OpenACC snippet

After ChatGPT generates the code, the system makes some notes about it. The observations written by ChatGPT are as follows:

In this code, we use the #pragma acc directive to specify which data should be copied to the GPU and which computations should be executed on the GPU. We also use a collapse clause to parallelize the two outer loops of the matrix multiplication.

Note that to run this code, you will need a GPU and a compiler that supports OpenACC. You will also need to link the OpenACC runtime library.

4.3 Calculate Jacobi interactive

To create the Jacobi interactive code, we asked ChatGPT the following question: ”Create Jacobi iterative algorithm C programming code for GPU processing using openACC.”.

In Listing 3 shows the answer with a part of the code using OpenACC.

```
#pragma acc data copyin(A[0:N][0:N],b[0:N]), copy(x[0:N]), create(x_new[0:N])
{
    for (k = 0; k < 100; k++) {
        #pragma acc kernels loop independent private(sum) collapse(2)
        for (i = 1; i < N-1; i++) {
            for (j = 1; j < N-1; j++) {
                sum = A[i][j-1] * x[j-1] + A[i][j+1] * x[j+1] +
                    A[i-1][j] * x[j] + A[i+1][j] * x[j] - A[i][j] * x[j] - b
                        [i];
                x_new[j] = sum / A[i][j];
            }
        }
        diff = 0.0;
        norm = 0.0;
        #pragma acc parallel loop reduction(max:diff,norm)
        for (i = 0; i < N; i++) {
            diff = fmax(diff, fabs(x_new[i] - x[i]));
            norm = fmax(norm, fabs(x_new[i]));
        }
        if (norm != 0.0 && diff / norm < TOLERANCE) {
            break;
        }
        #pragma acc parallel loop
        for (i = 0; i < N; i++) {
            x[i] = x_new[i];
        }
    }
}
```

Listing 3: A screenshot of the ChatGPT answer with C code from the OpenACC snippet

After ChatGPT generates the code, the system makes some notes about it. The observations written by ChatGPT are as follows:

In this code, we use the #pragma acc directive to specify which data should be copied to the GPU and which computations should be executed on the GPU. We also use a loop to perform Jacobi iterations to solve a system of linear equations.

Note that to run this code, you will need a compiler that supports OpenACC and a GPU. Additionally, you may want to adjust the number of iterations k or the tolerance TOLERANCE to obtain a more accurate solution.

4.4 Calculate prime number

To create the prime number code, we asked ChatGPT the following question: "Create prime number calculation C programming code for GPU processing using openACC.".

In Listing 4 shows the answer with a part of the code using OpenACC.

```
    #pragma acc data create(prime_numbers[0:LIMIT])
    {
        prime_numbers = (int*) malloc(LIMIT * sizeof(int));

        #pragma acc parallel loop reduction(+:count)
        for (int i = 2; i < LIMIT; i++) {
            if (is_prime(i)) {
                prime_numbers[count++] = i;
            }
        }
    }
```

Listing 4: A screenshot of the ChatGPT answer with C code from the OpenACC snippet

After ChatGPT generates the code, the system makes some notes about it. The observations written by ChatGPT are as follows:

In this code, the LIMIT constant is set to the upper limit for finding prime numbers.

The is_prime function checks if a given number is prime or not.

The prime_numbers array is allocated on the GPU using the acc data create directive. The acc parallel loop directive is used to parallelize the finding of prime numbers. The reduction clause is used to perform the reduction operation on the count variable across all threads.

The printf function is used to print the resulting prime numbers. Finally, the memory allocated on the GPU is freed using the free function.

5 Human-created codes

5.1 Calculate fractal Mandelbrot

Likewise to the code created by ChatGPT, we use the #pragma acc data directive to indicate the data that should be copied and run on the GPU. For code processing, we additionally employ the #pragma acc parallel directive.

In Listing 5 shows the answer with a part of the code using OpenACC.

```
#pragma acc data copy(counter)
#pragma acc parallel loop
for (i = 0; i < X_RESN; i++)
  for (j = 0; j < Y_RESN; j++)
  {

    z.real = z.imag = 0.0;
    c.real = ((float)j - 400.0) / 200.0;
    c.imag = ((float)i - 400.0) / 200.0;
    k = 0;

    do
    {

      temp = z.real * z.real - z.imag * z.imag + c.real;
      z.imag = 2.0 * z.real * z.imag + c.imag;
      z.real = temp;
      lengthsq = z.real * z.real + z.imag * z.imag;
      k++;
    } while (lengthsq < 4.0 && k < 1000);

    if (k == 1000){
      #pragma acc atomic update
        counter++;
    }
  }
```

Listing 5: A screenshot of the ChatGPT answer with C code from the OpenACC snippet

We employ the atomic directive to ensure that no two threads execute the operation contained inside at the same time, with the update option doing a combined reading and writing operation.

5.2 Calculate matrix multiplication

There are several approaches for creating matrix multiplication code; we utilize a simple and easy-to-implement technique.

In Listing 6 shows the answer with a part of the code using OpenACC.

```
#pragma acc data copyin (a[0:SIZE*SIZE], b[0:SIZE*SIZE]), copy(c[0:SIZE*SIZE])
{
    #pragma acc parallel loop gang vector collapse(2)
    for (i = 0; i < SIZE; ++i) {
      for (j = 0; j < SIZE; ++j) {
          a[i*SIZE+j] = (float)i + j;
          b[i*SIZE+j] = (float)i - j;
          c[i*SIZE+j] = 0.0f;
      }
    }
    #pragma acc parallel
    #pragma acc loop tile(256,256) independent
    for (i = 0; i < SIZE; ++i) {
      for (j = 0; j < SIZE; ++j) {
          temp  = 0.0;
                 #pragma acc loop reduction(+:temp)
                 for (k = 0; k < SIZE; ++k) {
                     temp += a[i*SIZE+k] + b[k*SIZE + j];
                 }
                 c[i*SIZE+j] = temp;
      }
    }
}
```

Listing 6: A screenshot of the ChatGPT answer with C code from the OpenACC snippet

As with ChatGPT code, we use the #pragma acc data directive to define the data to be copied and run on the GPU. A collapse clause is also used to parallelize the two external loops of matrix multiplication.

For code execution, ChatGPT utilized the #pragma kernels directive, and we used the #pragma parallel loop directive with some clauses. The tile clause can be used to improve the loop through enabling smaller blocks to operate to exploit data access. Another change we made was implementing the reduction operation in the last loop.

5.3 Calculate Jacobi interactive

The calculate Jacobi interactive is frequently used as an example in OpenACC programming, and there are several codes to execute it.

In Listing 7 shows the answer with a part of the code using OpenACC.

```
    #pragma acc data copy(A) create(Anew)
    while ( dt > MAX_TEMP_ERROR && iteration <= max_iterations ) {

        #pragma acc parallel loop
        for(i = 1; i <= ROWS; i++) {
            for(j = 1; j <= COLUMNS; j++) {
                Anew[i][j] = 0.25 * (A[i+1][j] + A[i-1][j] +
                                            A[i][j+1] + A[i][j-1]);
            }
        }

        dt = 0.0;

        #pragma acc parallel loop reduction(max:dt)
        for(i = 1; i <= ROWS; i++){
            for(j = 1; j <= COLUMNS; j++){
              dt = fmax( fabs(Anew[i][j]-A[i][j]), dt);
              A[i][j] = Anew[i][j];
            }
        }
```

Listing 7: A screenshot of the ChatGPT answer with C code from the OpenACC snippet

ChatGPT always attempts to utilize the #pragma kernels directive for code execution, whereas we attempt to use the #pragma parallel loop directive with some clauses.

We utilize the #pragma acc data directive, similar to the code generated by ChatGPT, to indicate the data that should be copied and run on the GPU. In both implementations, the loop was optimized employing the indentation operation to create the new matrix.

5.4 Calculate prime number

We employs the routine and parallel directives to run the code. To use function calls in parallel sections of the accelerator, the routine directive must be used, which asks the compiler to produce a device version of the function or subroutine so that it can be called from a region.

In Listing 8 shows the answer with a part of the code using OpenACC.

```
#pragma acc routine
int primo (long int n) {

        long int i;

    #pragma acc loop
        for (i = 3; i < (long int)(sqrt(n) + 1); i+=2)
             if (n%i == 0)
                 return 0;
        return 1;

}

    #pragma acc parallel loop reduction(+:quantidadePrimos)
    for (i = 3; i <= n; i += 2)
         if(primo(i) == 1) quantidadePrimos++;

    quantidadePrimos += 1;
```

Listing 8: A screenshot of the ChatGPT answer with C code from the OpenACC snippet

The parallel directive with a reduction before the second loop is used in the main program to calculate the number of prime numbers in the interval.

6 Experimental Setup

The tests were conducted on a server equipped with two Intel Xeon E5-2609 processors (1.7 GHz, 8 cores each, 20 MB cache), 128 GB of shared memory, and an NVIDIA GPU Tesla K80. The NVIDIA Tesla K80 is a dual-GPU system that employs two GK210B chipsets. This card features a total of 4992 CUDA cores clocked at 560 MHz, along with 24GB of GDDR5 vRAM, a 384-bit memory interface, and a 480 GB/s bandwidth.

All codes were compiled with the PGI Compiler 19.10 for optimal performance. Local, high-speed SSD (Solid-State Drive) drives were utilized to store the codes. The 64-bit Centos Linux distribution version 7.8 was utilized as the operating system.

7 Results

For analyzing the results, the codes were compiled using the PGI compiler and ran on the same server to ensure that the same computational resources were employed.

7.1 Calculate fractal Mandelbrot

For the Mandelbrot fractal calculation, we changed the width and height to compare the execution times of the codes generated by ChatGPT and the Human.

Table 1 shows the times following each execution. As can be observed that increasing the width and height of the code generated by ChatGPT increases the execution time.

As seen in Figure 3, as we increase the width and height of the code created by Hunan, the time fluctuation is very small; this is because to the implementation of the atomic directive

Table 1: Results of the Mandelbrot calculation code execution times

Matrix Size	Time (s)	
	ChatGPT	Human
1000 x 1000	0.560	0.224
3000 x 3000	0.195	0.220
7000 x 7000	0.347	0.225
10000 x 10000	0.542	0.254
12000 x 12000	0.811	0.225
15000 x 15000	1.106	0.231
17000 x 17000	1.625	0.239
20000 x 20000	2.137	0.244

in the code, which results in a significant speed advantage compared to code generated by ChatGPT.

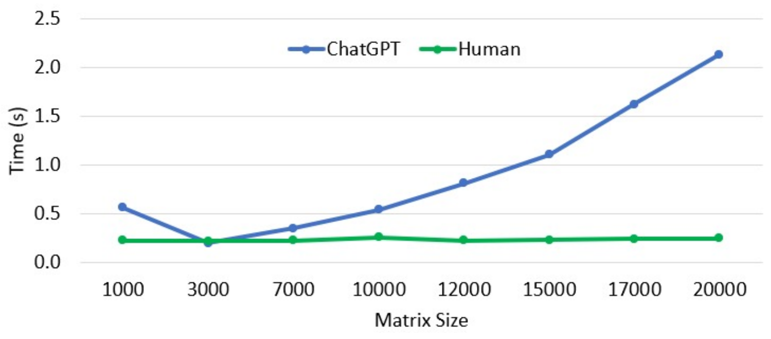


Figure 3: Results of the Mandelbrot calculation code execution times

7.2 Calculate matrix multiplication

The execution times of the code to calculate the Mandelbrot fractal and the code to calculate matrix multiplication, both developed using ChatGPT, rose as the matrix size increased, as shown in Table 2.

The execution times utilizing the Human-generated code remained low, which was feasible since we employed the reduction clause in the last loop. Figure 4 shows that increasing the matrix size greatly increases the execution time.

7.3 Calculate Jacobi interactive

The execution times of the codes generated by ChatGPT and the Human were close, with the code generated by the Human spending less time at times.

Table 3 shows the times obtained after running the codes. And both codes use the same directives, such as moving data to run on the GPU and the parallel directive.

Figure 5 depicts the outcomes of executing the codes; as previously stated, the results are near. The reduction clause was also used in both codes, ensuring that the time was kept low

Table 2: Results of the matrix multiplication calculation code execution times

Matrix Size	Time (s)	
	ChatGPT	Human
1024 x 1024	0.282	0.242
2048 x 2048	0.471	0.250
3072 x 3072	1.021	0.286
4096 x 4096	2.074	0.322
5120 x 5120	3.822	0.397
6144 x 6144	6.425	0.466
7168 x 7168	10.237	0.561
8192 x 8192	14.890	0.679

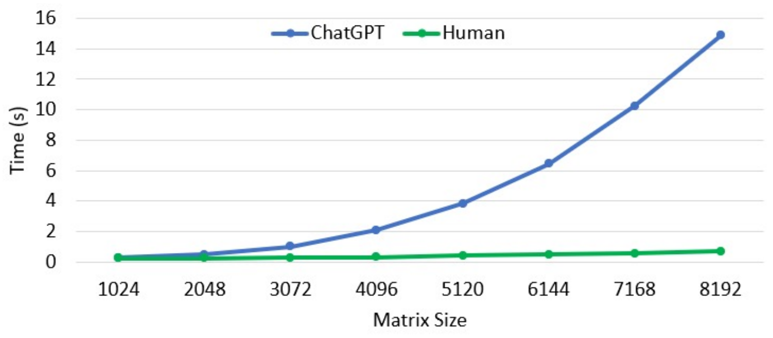


Figure 4: Results of the matrix multiplication calculation code execution times

Table 3: Results of the Jacobi calculation code execution times

Matrix Size	Time (s)	
	ChatGPT	Human
300 x 300	0.245	0.239
400 x 400	0.249	0.228
500 x 500	0.233	0.247
600 x 600	0.239	0.241
700 x 700	0.248	0.238
800 x 800	0.269	0.240
900 x 900	0.275	0.288
1000 x 1000	0.272	0.248

even when the size of the matrix was changed.

7.4 Calculate prime number

When looking at the execution time of the code created by ChatGPT, we observe that it behaves similarly to the times to calculate the Mandelbrot fractal and matrix multiplication, in that as the N size increases, the time increases almost equally and works with a scale that is linear as

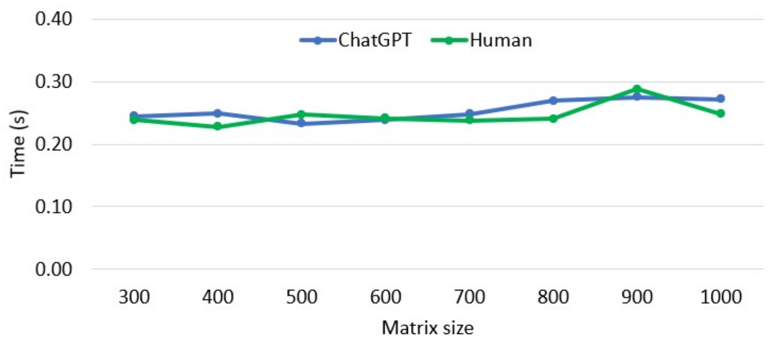


Figure 5: Results of the Jacobi calculation code execution times

Table 4: Results of the prime number calculation code execution times

N Size	Time (s)	
	ChatGPT	Human
1000000	0.245	0.239
2000000	0.249	0.228
3000000	0.233	0.247
4000000	0.239	0.241
5000000	0.248	0.238
6000000	0.269	0.240
7000000	0.275	0.288
8000000	0.272	0.248

can be seen in Table 4.

Even when the N size was increased, the execution times for the human-generated code remained low. The use of the routine and parallel directives increased the code gain significantly greater than that generated by ChatGPT. Figure 6 show the results obtained with the two codes.

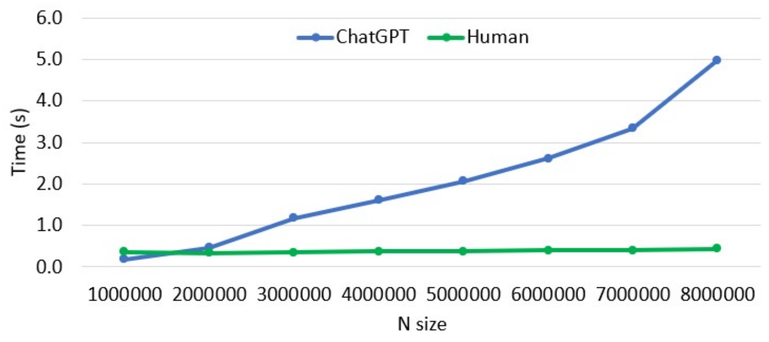


Figure 6: Results of the prime number calculation code execution times

8 Conclusions

Artificial intelligence is being employed in a variety of fields, with several different kinds of applications available. In education, as in other fields, different applications that implement AI have been developed.

With the arrival of ChatGPT, many people have become interested in how they can employ AI to help them especially with their tasks. Based on studies, AI is already being employed by students in their academic tasks.

This study looks at how ChatGPT can help students with some of these activities and how effective it is. We asked ChatGPT to generate three parallel C programming codes for GPUs using OpenACC, and then we will write the same codes using various approaches to achieve the best performance in each one.

The time for calculating the Mandelbrot fractal increased as we increased the size of the height and width using the code generated by ChatGPT, in contrary to the behavior of the code generated by the Human, which remained close, that is, with little variation between values.

When running the code to calculate matrix multiplication, it had the same behavior as the code for calculating the Mandelbrot fractal. The results presented were caused by of the Human's improved usage of OpenACC directives and clauses in both codes.

However, for the Jacobi method calculation, the values of the execution times were in for both the generated codes, ChatGPT and Human. The best OpenACC directives and clauses were used for execution in both cases.

The Human-generated codes showed to be more efficient in all environments, even when the values in the Jacobi method calculation were very close. This is a consequence of the fact that we can employ OpenACC directives and clauses more effectively.

In all cases, the codes generated by ChatGPT work as expected, and the applications run without issues. Overall, the codes make efficient utilization of OpenACC directives and clauses and ChatGPT can be utilized to generate code without serious issues. After generating the code, we can make changes to improve its performance.

Acknowledgment

The authors are grateful to the Computer Institute of Rio de Janeiro Federal University for providing the computing resources utilized to conduct the experiments described in this research.

References

[1] Ömer Aydın and Enis Karaarslan. Is chatgpt leading generative ai? what is beyond expectations? january 2023.

[2] Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large language models in machine translation. pages 858–867, jun 2007.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

[4] Shaohao Chen. Introduction to openacc. *Research Computing Services Information Services and Technology Boston University*, 2017.

[5] Grant Cooper. Examining science education in chatgpt: An exploratory study of generative artificial intelligence. *Journal of Science Education and Technology*, pages 1–9, 03 2023.

[6] Djoerd Hiemstra. *Language Models*, pages 1591–1594. Springer US, Boston, MA, 2009.

[7] Jeff Larkin. Introduction to openacc. *NVIDIA*, 2018.

[8] Chung Kwan Lo. What is the impact of chatgpt on education? a rapid review of the literature. *Education Sciences*, 13(4), 2023.

[9] Regina Luttrell, Adrienne Wallace, Christopher McCollough, and Jiyoung Lee. The digital divide: Addressing artificial intelligence in communication education. *Journalism & Mass Communication Educator*, 75(4):470–482, 2020.

[10] Fadel M. Megahed, Ying-Ju Chen, Joshua A. Ferris, Sven Knoth, and L. Allison Jones-Farmer. How generative ai models such as chatgpt can be (mis)used in spc practice, education, and research? an exploratory study. 2023.

[11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013.

[12] John V. Pavlik. Collaborating with chatgpt: Considering the implications of generative artificial intelligence for journalism and media education. *Journalism & Mass Communication Educator*, 78(1):84–93, 2023.

[13] Gabriel P. Silva, Calebe P. Bianchini, and Evaldo B. Costa. *Programação Paralela e Distribuída com MPI, OpenMP e OpenACC para computação de alto desempenho*. Casa do Codigo, 2021.

[14] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model, 2022.

[15] Longyue Wang, Chenyang Lyu, Tianbo Ji, Zhirui Zhang, Dian Yu, Shuming Shi, and Zhaopeng Tu. Document-level machine translation with large language models, 2023.

[16] Olaf Zawacki-Richter, Victoria I. Marín, Melissa Bond, and Franziska Gouverneur. Systematic review of research on artificial intelligence applications in higher education – where are the educators? *International Journal of Educational Technology in Higher Education*, 16(1):39, 12 2019.