Domain Specific Language for high performance computing

Ryuki Hiwada s1280076 Supervised by Naohito Nakasato

1 Abstract

2 Introduction

2.1 Background

In astrophysics and astronomy, to numerically calculate the dynamical evolution of N particles interacting gravitationally, N-body simulations are required. Figure 1 shows the equation for interparticle interactions in N-body simulations. If the equation is naively computed, the time complexity of calculation of interparticle interactions is $O(N^2)$, where N is the number of particles. Therefore, parallelization is required to speed up numerical simulations. To write a parallelized code for a numerical simulation, a user needs to understand the architecture of computer systems in detail. If a parallelized code is automatically generated by only describing the formulas and data of the numerical simulation, the above problems are solved. To realize the parallelization, we will develop Domain Specific Language (DSL), which is a programming language specialized to a domain, for example, SQL and HTML.

2.2 parallelization

SIMD is one of the categories in Flynn's taxonomy 1, related to computer arithmetic processing. Using SIMD, arithmetic operations can be applied simultaneously to multiple data. In this paper, we compute and parallelize multiple particles. For example, as shown in Figure 1, the computation of acceleration for different particles is independent of each other, allowing the process to be parallelized. As a result, for instance Assuming that the variables for particles are double-precision floating-point numbers, we can execute an operation on four elements simultaneously using the AVX2 instruction, it is possible to perform operations on four elements. Therefore, compared to non-parallelized code, there is a potential to accelerate the computation by four times. We will describe the design of a DSL in the next section to generate such parallelized code from the formulation of particle-particle interactions.

2.3 Overview of DSL

We have created a Domain Specific Language (DSL)named (DSL name). This language's processor can read code describing particle interaction formulas and generate code that accelerates these computations. The language allows for the definition of variables and the description of interaction formulas. In the variable definition section, it's possible to define variables such as the mass and position of particles, variables for storing results, and other necessary variables. For the formula part, we have enabled calculations including basic arithmetic operations, sqrt, power functions, and computing the norm of vectors. From this description, a kernel function is generated, which can then be called and used. We have created a Domain Specific Language (DSL)named (DSL name). This language's processor can read code describing particle interaction formulas and generate code that accelerates these computations. The language allows for the definition of variables and the description of interaction formulas. In the variable definition section, it's possible to define variables such as the mass and position of particles, variables for storing results, and other necessary variables. For the formula part, we have enabled calculations including basic arithmetic operations, sqrt, power functions, and computing the norm of vectors. From this description, a kernel function is generated, which can then be called and used.

2.4 Aim of this paper

3 implementation

We use Sympy, a library in Python for implementing the implementation of our (DSL). SymPy supports various operations, and the formulas using these operations are internally treated as syntax trees. We leveraged this functionality to read formulas and generate code capable of parallel execution. For parallelization, we used an instruction set named Advanced Vector Extensions 2 (AVX2). AVX2 is one of the extended instruction sets implemented in Intel's CPUs. The next section will explain how parallelization is carried out using AVX2.

3.1 AVX2 for parallelization

To execute SIMD operations, SIMD-specific registers are used, and multiple data are computed simultaneously to match the bit size of the registers. For example, in the case of CPUs that support AVX2 instructions, the registers are 256 bits, allowing four double-precision floating-point numbers to be stored and computed at the same time. To explicitly handle SIMD instructions in high-level programming languages like C and C++,

intrinsic functions are used. Additionally, to handle SIMD-specific registers, specialized types are used. For C++, the file immintrin.h, which is available as a standard in the language, is included to handle these. As an example of parallelization, the paper explains the parallelization of the gravitational interaction formula shown in Figure 1.

3.2 design of (name of DSL)

```
1 #include <iostream>
2
3 int main(){
4     std::cout << "Hello, world!" << std::endl;
5     return 0;
6 }
```

3.3 implementation of (name of DSL)

3.4 Use Sympy for DSL development

4 Conclusion

5 Acknowledgement